A NUMERICAL STUDY OF KdV SOLITONS USING CUBIC SPLINE APPROXIMATION

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

ANWARUL HASAN

to the

DEPARTMENT OF NUCLEAR ENGINEERING & TECHNOLOGY

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JANUARY, 1987

DEDICATED

TO

MY PARENTS

AND

TEACHERS

NET-1987-M-HAS--NUM

CERTIFICATE

This is to certify that the thesis entitled, "A NUMERICAL STUDY OF KdV SOLITONS USING CUBIC SPLINE APPROXIMATION" by ANWARUL HASAN is a record of work carried out under my supervision and has not been submitted elsewhere for a degree.

January, 1987.

M.S Kalra

(M.S. Kalra)
Assistant Professor
Nuclear Engineering and Technology Programme
Indian Institute of Technology
Kanpur-208016

ACKNOWLEDGEMENTS

I am highly grateful to my guide Dr. M.S. Kalra for his invaluable guidance, timely help and encouragement throughout my thesis work. It was a result of his affection and understanding that I could work to the best of my abilities. He commands a great respect and regard in my heart.

I wish to thank Mrs. Kalra for her patience and co-operation extended to my guide, particularly in last days of my thesis completion.

My thanks are due to my teachers Dr. K. Sri Ram, Mr. P. Munshi and Dr. A. Sengupta for their help to me and my classmates.

I would also like to thank all the other department staff - Mr. Pathak, Mr. Tomar, Mr. Yadav, Mr. Tripathi and Mr. Gopal for their cooperation and help.

I take this opportunity to express sincere thanks and praise to all my friends specially Br. Asim and Mr. Manoj for creating a congenial atmosphere and making my stay at I.I.T. Kanpur enjoyable and memorable.

And finally, I would like to acknowledge the contribution of Mr. U.S.Mishra for typing this thesis.

-ANWARUL HASAN

CONTENTS

		Page
	CERTIFICATE	ii
	ACKNOWLEDGEMENTS	iii
	LIST OF FIGURES	vi
	LIST OF TABLES	vii
	ABSTRACT	ix
CHAPTER 1	INTRODUCTION	
	1.1 Foreword	1
	1.2 The KdV Equation and Its Transformations	3
	1.3 Physical Systems Leading to KdV Equation	5
	1.3.1 Shallow Liquid Waves	5
	1.3.2 Ion Plasma Waves and the KdV Equation	6
	1.4 Literature Review	9
	1.5 Outline of the Present Work	11
CHAPTER 2	ANALYTICAL METHODS FOR THE KdV EQUATION	
	2.1 Single-Soliton Solutions for the KdV Equation	13
	2.2 Spectral Transform Method for the KdV Equation	15
CHAPTER 3	NUMERICAL METHODS APPLICABLE TO THE KdV Equation	
	3.1 Introduction	19
	3.2 Basic Numerical Methods	19
	3.3 The Finite Difference Methods	21
	3.4 Finite Element Methods	22
	3.4.1 The Collocation Method	24
	3.4.2 The Least Square Method	24
	3.4.3 The Galerkin Method	25
	3.5 A Comparison of Different Methods	25
	3.6 Convergence, Consistency and Stability	26
CHAPTER 4	NUMERICAL COMPUTATIONS AND RESULTS	
	4.1 Preliminary Remarks	28
	4.2 Calculation of Galerkin Constants	28
	4.3 Initial Conditions	32
	4.4 Solution of ODE System	33
	4.5 Evaluation of the Cubic Spline for the Final Solution	34
	4.6 Results and Their Accuracy	34

Page

CHAPTER 5	COMPARISON WITH PREVIOUS WORKS AND CONCLUSIONS	
	5.1 Comparison of L_∞ and L_2 Errors	47
	5.2 Conclusion and Suggestions	48
APPENDIX A	TRANSFORMATIONS FOR KdV EQUATION	55
APPENDIX B	LISTING OF THE COMPUTER CODE	56
APPENDIX C	DETAILED TABLES FOR CALCULATED SOLUTIONS	78
REFERENCES		87

LIST OF FIGURES

Figure No.	Title	Page
2.1	A Plot of the KdV Soliton [Eq.2.10 with d = 0] at T = 0.	16
4.1	An Outline of the Present Computer Code	38
4.2	Initial and Calculated/Expected Positions of KdV Soliton at T = 2.0 Second (N = 40)	39
4.3	Initial and Calculated/Expected Positions of KdV Soliton at T = 1.2 Second (N = 50).	40

LIST OF TABLES

Table	Title	Page
4.1	Comparison of Actual Initial Data and Its Cubic Spline Representation (N = 40).	41
4.2	Errors in the Observed Locations of the Soliton Peaks (N = 40)	42
4.3	Errors in the Observed Locations of the Soliton Peaks (N = 50)	43
4.4	A Detailed Comparison of the Calculated and Expected Single-soliton Solutions (T = 1.2 second, N = 50).	44
4.5	Errors in Peak Values for Different Evolution Times (T) (N = 40)	45
4.6	Errors in Peak Values for Different Evolution Times (T) (N = 50)	46
5.1	Comparison with Alexander-Morris Galerkin Schemes (N = 40)	50
5.2	Comparison with Alexander-Morris Galerkin Schemes (N = 50 or 60)	51
5.3	Comparison with Finite-Difference Schemes (N = 40)	52
5.4	Comparison with Petrov-Galerkin Schemes	53
5.5	Computation Time for Different Runs	54
C1	Calculated Single-Soliton Solutions for N = 40, T = 0.0	78

Page	Title	Page
C2	Calculated Single-Soliton Solutions for N = 40, T = 0.5	79
C3	Calculated Single-Soliton Solutions for N = 40, T = 1.0	80
C4	Calculated Single-Soliton Solutions for N = 40, T = 1.5	81
C5	Calculated Single-Soliton Solutions for N = 40, T = 2.0	82
C6	Calculated Single-Soliton Solutions for N = 50, T = 0.0	83
C7	Calculated Single-Soliton Solutions for N = 50, T = 0.4	84
C8	Calculated Single-Soliton Solutions for N = 50, T = 0.8	85
C9	Calculated Single-Soliton Solutions for N = 50, T = 1.2.	86

ABSTRACT

A numerical solution for the KdV solitons has been obtained using Galerkin scheme. The basis functions used are cubic splines. The scheme gives a system of ordinary differential equations in the time variable. The solution in time is evolved using IMSL library subroutines.

The solution obtained is compared with the various finite difference and finite element schemes used previously. The errors are found to be significantly less as compared with the finite difference schemes reported in the literature. The Galerkin scheme used here is also found to give better results than the Petrov-Galerkin and the Modified Petrov-Galerkin schemes reported in recent literature.

The results obtained here also appear to be generally more favorable than the dissipative Galerkin schemes based on cubic splines and quintic boundary basis functions used previously.

CHAPTER 1

INTRODUCTION

1.1 Foreword

It is well known that the initial and boundary value problems associated with nonlinear partial differential equations are very difficult to handle in a general way. Some specific problems have been tackled from time to time by methods specifically suited to the individual problems.

Over the past two decades, a remarkable development in our understanding of certain classes of nonlinear partial differential equations known as (nonlinear) evolution equations has taken place. A fundamental relationship between some of these nonlinear equations and ordinary linear differential equations of Sturm-Liouville type has been demonstrated through a variety of Spectral Transform. Even at our present level of understanding, the basis for such a fundamental relationship remains somewhat obscure [1].

The main reasons of interest in these nonlinear evolution equations are two fold:

1. These arise in a natural way in a large number of physical problems.

2. They possess special type of solutions which can be of great practical use.

These special solutions take the form of localized travelling disturbances, or pulse, with peculiar properties.

1. Their speed, (effective) width and amplitude are inter-related unlike the solutions of linear partial differential equations which have travelling solutions.

2. Their shape does not get distorted as they travel through the medium.

3. They retain their shape, amplitude and speed (i.e. complete identity) even after interactions among themselves, and suffer only a phase shift upon such interactions.

Due to above mentioned particle-like properties, such localized travelling disturbances have come to be known as Solitons.

Our interest in the present work is in the numerical simulation studies of single-soliton solutions of one such evolution equation known as Korteweg-deVries (KdV) equation. The KdV equation arises in a number of hydrodynamic problems including problems relevent to plasma physics.

In the remainder of this chapter, we introduce the KdV equation and briefly discuss some physical situations leading to the KdV equation. This is followed by a brief review of the literature and an outline of the present work.

1.2 The Korteweg-deVries (KdV) Equation and Its Transformations

The KdV equation may be written as:

$$A \frac{\partial u}{\partial t} + Bu \frac{\partial u}{\partial x} + C \frac{\partial^3 u}{\partial x^3} = 0, \qquad (1.1)$$

where $u \equiv u(x,t)$, x and t are independent variables, and A, B and C are all real, nonzero constants. Upon division by A, one can write Eq.(1.1) as:

$$\frac{\partial u}{\partial t} + au \frac{\partial u}{\partial x} + b \frac{\partial^3 u}{\partial x^3} = 0 , \qquad (1.2)$$

where a = B/A and b = C/A. It should be pointed out that suitable transformations of u, x and t allow the introduction of arbitrary constants multiplying the three terms in Eq.(1.1) or (1.2). For example, a rescaling of Eq.(1.2) with

$$u \rightarrow u/a$$

will yield:

$$u_t + uu_x + \varepsilon u_{xxx} = 0 , \qquad (1.3)$$

where $\varepsilon = b$. In writing Eq.(1.3) we have used a standard notation for the partial derivatives, viz.:

$$u_t \equiv \frac{\partial u}{\partial t}, \quad u_x \equiv \frac{\partial u}{\partial x} \text{ and } u_{xxx} \equiv \frac{\partial^3 u}{\partial x^3}$$

A rescaling with

$$x \rightarrow x\, b^{1/3}, \text{ and}$$
$$u \rightarrow u\, a^{-1}\, b^{1/3}$$

will give coefficients of unity in front of each terms, i.e.,

$$u_t + uu_x + u_{xxx} = 0 . \qquad (1.4)$$

Instead, a transformation $x \to xb^{1/3}$ and $u \to -\frac{1}{6} u a^{-1} b^{1/3}$ in Eq.(1.2) or a rescaling with $u \to -\frac{1}{6} u$ in Eq.(1.4) will yield another useful form of the KdV equation:

$$u_t - 6uu_x + u_{xxx} = 0. \qquad (1.5)$$

As will be seen in Section 2.2, the form (1.5) above is most convenient for a discussion of the Spectral Transform Method.

It may be mentioned that some of these transformations may not be necessarily unique. For example, in Appendix A, we give two different transforms which change Eq.(1.1) to Eq.(1.14).

It is perhaps worthwhile to point out that if $\varepsilon < 0$ in Eq.(1.3), a transformation $u \to -u$ and $x \to -x$ will lead to a positive coefficient of the term u_{xxx}.

In the present work we would be interested in the solution of the KdV equation for localized initial data $u(x,0)$, where $u(x,0)$ and its derivatives with respect to x vanish sufficiently rapidly as $|x| \to \infty$. It is also of interest to note that without the nonlinear term, uu_x, Eq.(1.3) describes linear dispersive waves, whereas, without the dispersive term, u_{xxx}, the same equation describes the well known shock phenomena in fluid dynamics [2,3].

1.3 Physical Systems Leading to the KdV Equation

As has been remarked earlier a number of physical systems, mostly related to hydrodynamics, lead to the KdV equation. Unfortunately, somewhat tedious perturbation calculations are needed before the equation of interest, such as the KdV equation, is finally obtained. We briefly mention here two examples.

1.3.1 Shallow Liquid Waves

We consider the motion of plane disturbance on the surface of a liquid in a channel of constant depth.
Let,

h = the constant depth of the liquid in the channel

a = the amplitude of the disturbance, and

l = the effective width of the disturbance along the channel.

If one assumes that

$$\frac{a}{h} << 1, \text{ and}$$

$$\frac{h}{l} << 1,$$

a perturbation development of the resulting fluid motion can be obtained. If the fluid motion is assumed to be nonviscous, incompressible, irrotational, and bounded below by a hard horizontal bed, then the following equation for the resulting motions is obtained [1,4]:

$$\eta_t + c_o \eta_x + \frac{3}{2} \frac{c_o}{h} \eta\eta_x + \frac{1}{2} c_o \sigma \eta_{xxx} = 0 \qquad (1.6)$$

where

$\eta \equiv \eta(x,t)$, the boundary of the free surface,

$c_o^2 = gh$,

g = acceleration due to gravity,

$\sigma = \frac{h^2}{3} - \frac{T}{\rho g}$,

T = surface tension, and

ρ = density of the fluid.

If the surface tension is negligible, i.e. $T << h^2 \rho g$, the quantity $\frac{1}{2} c_o \sigma$ in Eq.(1.6) simplifies to $\frac{1}{6} c_o h^2$. [1] .

It is now straightforward to see that the Galilean transformation

$$x \rightarrow x + c_o t$$

transforms Eq.(1.6) to the KdV Eq.(1.2).

1.3.2 Ion Plasma Waves and the KdV Equation

In this section we will show that nonlinear ion-acoustic waves travelling with a speed close to the linear acoustic speed in a cold plasma ($T_i = 0$) obey the KdV equation and thus possess the soliton characteristics consider the simple case of one-dimensional disturbance. The fluid equations of motion and continuity are [5,6,7]

$$\frac{\partial v_i}{\partial t} + v_i \frac{\partial v_i}{\partial x} = - \frac{e}{m} \frac{\partial \varphi}{\partial x} \qquad (1.7)$$

$$\frac{\partial n_i}{\partial t} + \frac{\partial}{\partial x}(n_i v_i) = 0 \qquad (1.8)$$

where

v_i = speed of the ions,

e = magnitude of electron charge,

m = electron mass,

φ = electric potential,

n_i = ion density, and,

x and t denote the space and time variables respectively. If we neglect the electron inertia, the Poisson's equation takes the form [5]

$$\varepsilon_o \frac{\partial^2 \varphi}{\partial x^2} = e(n_o e^{e\varphi/KTe} - n_i) \qquad (1.9)$$

where

ε_o = permittivity of vacuum,

n_o = unperturbed electron density

K = Boltzmann constant

Te = electron temperature, and

other variables are as in Eqs.(1.7) and (1.8).

The following dimensionless variables make all the coefficients unity:

$$x' = x/\lambda_D$$

$$t' = \Omega_p\, t$$

$$X = e\varphi/K\,Te$$

$$v' = v_i/v_s$$

$$n' = n_i/n_o$$

where

λ_D = Debye length = $\left(\frac{\varepsilon_o K T_e}{n_o e^2}\right)^{1/2}$

Ω_p = plasma ion frequency = $\left(\frac{n_o e^2}{\varepsilon_o M}\right)^{1/2}$

v_s = acoustic speed ($T_i = 0$)

$= \left(\frac{K T_e}{M}\right)^{1/2}$,

M = ion mass.

The Eqs.(1.7) to (1.9) now become:

$$\frac{\partial v'}{\partial t'} + v' \frac{\partial v'}{\partial x'} = - \frac{\partial X}{\partial x'} \qquad (1.10)$$

$$\frac{\partial n'}{\partial t'} + \frac{\partial}{\partial x'}(n'v') = 0 \qquad (1.11)$$

$$\frac{\partial^2 X}{\partial x'^2} = e^X - n' \qquad (1.12)$$

Now, n', X and v' can be expanded in terms of a dimensionless amplitude parameter :

$$\begin{aligned} n' &= 1 + \delta n_1 + \delta^2 n_2 + \ldots \\ X &= \delta X_1 + \delta^2 X_2 + \ldots \\ v' &= \delta v_1 + \delta^2 v_2 + \ldots \end{aligned} \qquad (1.13)$$

We also transform the variables x' and t' to ξ and τ through 5

$$\xi = \delta^{1/2} (x' - t') \qquad (1.14)$$
$$\tau = \delta^{3/2} t'$$

So that

$$\frac{\partial}{\partial t'} = \delta^{3/2} \frac{\partial}{\partial \tau} - \delta^{1/2} \frac{\partial}{\partial \xi} \qquad (1.15)$$
$$\frac{\partial}{\partial x'} = \delta^{1/2} \frac{\partial}{\partial \xi}$$

If we now assume that n_1, ξ_1, v_1 tend to zero as $|\xi| \to \infty$, it can be shown with the help of Eqs.(1.10) to (1.15) that [5] :

$$n_1 = X_1 = v_1 = U \qquad (1.16)$$

where U satisfies the KdV equation:

$$\frac{\partial U}{\partial \tau} + U \frac{\partial U}{\partial \xi} + \frac{1}{2} \frac{\partial^3 U}{\partial \xi^3} = 0 \qquad (1.17)$$

Thus ion waves of amplitude one order higher than linear are described by the KdV equation.

.4 Literature Review

The soliton was discovered and named in 1965 by Zabuskey and Krauskal [8], who were experimenting with the numerical solution of the KdV equation. The KdV equation

had been introduced earlier at the end of the last century to describe wave motion in shallow canals by D.J. Korteweg and G. de Vries (see Section 1.3.1). The first scientific description of soliton as a natural phenomenon goes however back to first half of the nineteenth century and was reported by J. Scott-Russel in 1844 [4,9,10,11].

But the real breakthrough occurred in 1967 when the idea of spectral transform technique was introduced by Gardener, Green, Kruskal and Miura as a means to solve the Cauchy problem for the KdV equation [12] . Soon afterwards Lax put the method into a framework, that provided a clear indication of its generality and greatly influenced future developments. A few years later, in 1971, Zakharov and Shabat, by a nontrivial extension of previous approaches solved the Cauchy problem for another important nonlinear evolution equation, so-called nonlinear Schroedinger equation [9,13] . The way was thereby opened for the search and discovery of several other nonlinear evolution equations, or rather classes of such equations, solvable by these techniques. This process continues unabated to the present day. The applications of this subject have been percolating through the whole of physics (from nonlinear optics to hydrodynamics, from plasma to elementary particle physics, from lattice dynamics to electrical networks, to superconductivity, to cosmology and to epidermiology etc).

This is of course related to the central role played by non-linear evolution in mathematical physics, and more generally in applied mathematics. In fact the spectral transform approach (see Section 2.2) constitute in some sense an extension to a nonlinear context of the Fourier transform technique, whose all-pervading role for solving and investigating linear phenomena is well known.

We will now briefly outline the numerical simulation studies for nonlinear evolution equations reported in literature. As has been mentioned above Zabusky and Kruskal [8] were the first to study KdV equation through a finite difference method. They used a leapfrog scheme (see Section 3.2). An improved scheme was later obtained by Grieg and Morris [14] . This was a hopscotch scheme. Fornberg and Witham [25] devised an accurate scheme using spectral methods for x-variable and leapfrog in t.

Among the finite element methods, Wahlbin [15] suggested a dissipative Galerkin scheme. Other workers who used Galerkin or Petro-Galerkin schemes are Andersen and Michell [26], Alexander and Morris [16] , Winthev, Mitchell and Schoombie [17] and Sanz-Serna and Chriesti [18].

1.5 Outline of the Present Work

In Chapter 2 we describe some analytical methods to obtain soliton solutions of the KdV equation. In Section 2.1 a method to construct single-soliton solutions is described,

whereas in Section 2.2 a more general technique (the Spectral Transform Method) is briefly outlined.

In Chapter 3 the two main numerical methods, viz. the finite difference and the finite element techniques are briefly outlined for partial differential equations in general and for the KdV equation in particular. This is followed by the actual numerical computations and their results in Chapter 4. The results obtained during this work are compared in detail with the previously available results in Chapter 5.

CHAPTER 2

ANALYTICAL METHODS FOR THE KdV EQUATION

2.1 Single-Soliton Solutions for the KdV Equation

In order to obtain single-soliton solutions of the KdV equation we seek travelling solution of permanent shape and size by trying solutions of the form

$$u(x,t) = U(\xi), \tag{2.1}$$

where

$$\xi = x - ct. \tag{2.2}$$

As a result we have,

$$\begin{aligned} u_t &= -c\ U_\xi \\ u_x &= U_\xi \\ u_{xxx} &= U_{\xi\xi\xi}\ . \end{aligned} \tag{2.3}$$

Substituting Eqs.(2.1) to (2.3) in Eq.(1.3) we get the ordinary differential equation

$$-c\ U_\xi + U\ U_\xi + \varepsilon\ U_{\xi\xi\xi} = 0, \tag{2.4}$$

which can be integrated once with respect to ξ to yield

$$-c\ U + \frac{1}{2} U^2 + \varepsilon\ U_{\xi\xi} = A, \tag{2.5}$$

where A is a constant of integration. Multiplying Eq.(2.5) by U_ξ and integrating we get,

$$-\frac{1}{2} c U^2 + \frac{1}{6} U^3 + \frac{1}{2} \varepsilon U_\xi^2 = AU + B \qquad (2.6)$$

For soliton solutions we use the boundary conditions, $U, U_\xi, U_{\xi\xi} \to 0$ as $|\xi| \to \infty$. Therefore, we have

$$A = B = 0$$

With these values of A and B and with a slight rearrangement, Eq.(2.6) can be written as

$$U_\xi^2 = \frac{U^2}{3\varepsilon} \cdot (3c - U) \qquad (2.7)$$

taking square roots of both sides in Eq.(2.7), the dependent variables U and the independent variable ξ may be separated as follows:

$$d\xi = \frac{\sqrt{3\varepsilon}\, dU}{U\sqrt{3C - U}} \qquad (2.8)$$

The last equation may be integrated using standard methods to yield

$$U(\xi) = 3c \ \mathrm{Sech}^2 \left(\frac{1}{2} \sqrt{\frac{c}{\varepsilon}}\, \xi + d\right), \qquad (2.9)$$

where c is a constant ≥ 0 and d is an arbitrary constant.

By substituting for ξ from Eq.(2.2) we obtain

$$u(x,t) = 3c \ \mathrm{Sech}^2 \left(\frac{1}{2} \sqrt{\frac{c}{\varepsilon}} (x - ct) + d\right) \qquad (2.10)$$

This is a single-soliton solution of the KdV equation, which we have been seeking. It can be seen that the amplitude

width and speed are all determined by a single parameter c, and are thus interdependent. In particular the pulse height, width and speed are proportional to c, $c^{-1/2}$, and c respectively.

Lastly we may remark that if we put $\varepsilon = 1$ in Eq.(2.10) we get the single-soliton solution for the KdV Eq. (1.4):

$$u(x,t) = 3c\ \mathrm{Sech}^2\left(\tfrac{1}{2}\sqrt{c}\,(x - ct) + d\right) \qquad (2.11)$$

Furthermore, in Eqs.(2.10) and (2.11) if we take d = 0, the peak of the soliton coincides with x = 0 at t = 0. A plot of Eq.(2.10) for d = 0 and c = 1 at t = 0 is shown in Fig.(2.1) for two different values of ε .

2.2 Spectral Transform Method for the KdV Equation

In this section we briefly discuss Schroedinger spectral transform which is appropriate for solving a class of nonlinear evolution equations of which KdV equation is a special case. For this purpose we consider the stationary Schroedinger equation:

$$-\Psi_{xx}(x,k) + u(x)\,\Psi(x,k) = k^2\,\Psi(x,k) \quad -\infty < x < +\infty \qquad (2.11)$$

Here u(x) is a given real function that vanishes sufficiently fast asymptotically, say

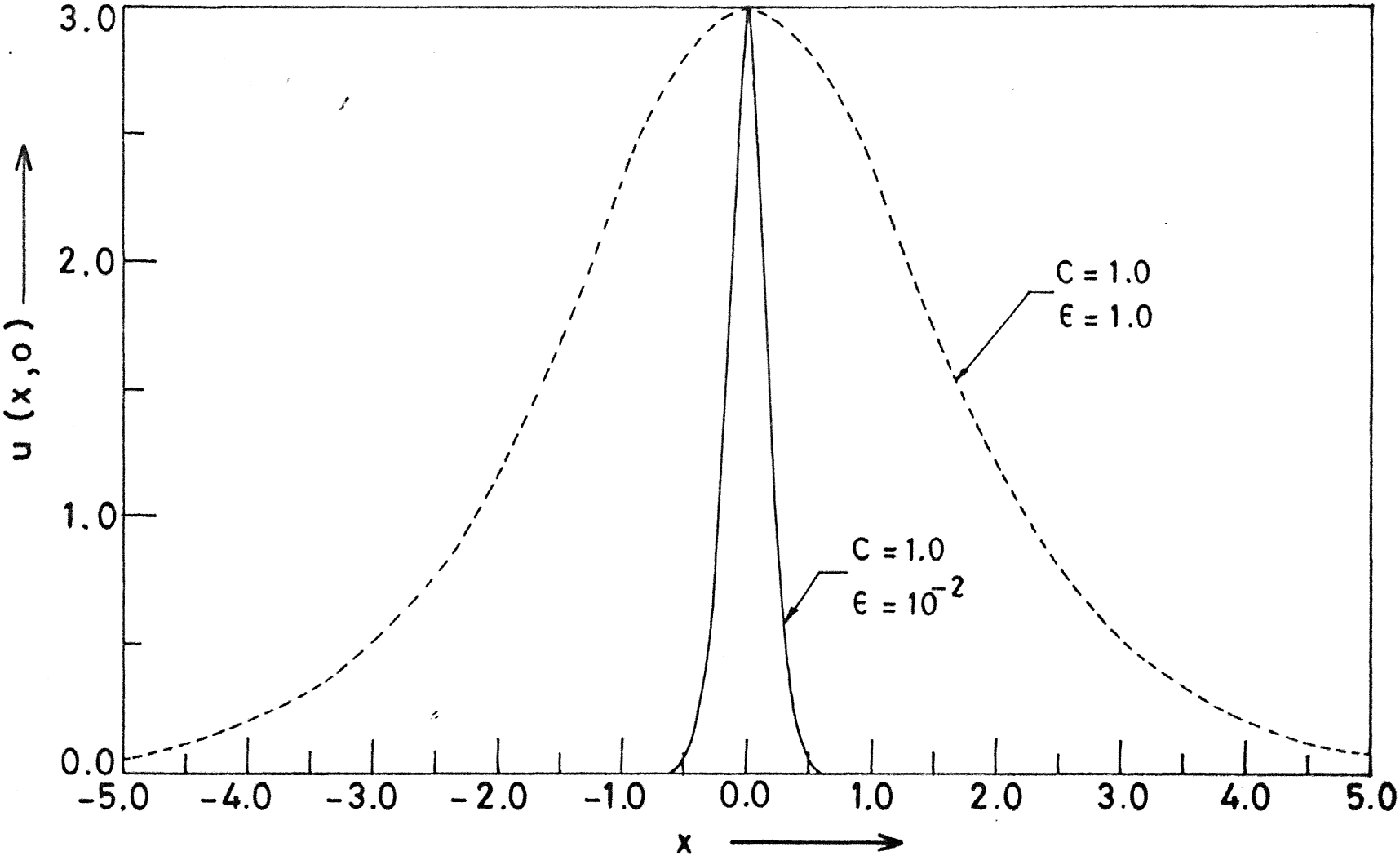

FIG. 2.1 A PLOT OF A KdV SOLITON Eq·2.10 WITH d=0 AT T=0

Eq.(2.11) is a singular Sturm-Liouville problem. The spectrum of the Sturm-Liouville problem characterized by Eq.(2.11) consists of two components:

1. A continuum, including all positive values of the eigen value, k^2.

2. A number of discrete negative eigen values $k^2 = - p_n^2$, $p_n > 0$, $n = 1,2,....,N$.

The continuum part of the spectrum is completely characterized by the reflection coefficients $R(k)$ [9,19] . On the other hand to completely characterize the discrete part one needs not only the values of p_n but also the corresponding normalization coefficients given by:

$$\rho_n = \left[\int_{-\infty}^{+\infty} dx\, f_n^2(x) \right]^{-1} , \quad n = 1,2,...,N \tag{2.13}$$

where

$f_n(x)$ is the eigen function corresponding to p_n.

The spectral transform S of the function $u(x)$ is by definition the collection of data

$$S(u) = \{ R(k), -\infty < k < +\infty ; \rho_n , \ n = 1,2,....,N \} \tag{2.14}$$

It is found that there is $\overset{\text{one-to-one}}{\longleftrightarrow}$ correspondence between functions $u(x)$ and the spectral transform $S[u]$. If u also depends on another variable say t, i.e., $u \equiv u(x,t)$, then we get a time dependent spectral transform:

$$u(x,t) \rightarrow S(t) \tag{2.15}$$

The all-important, and highly nontrivial discovery of the last two decades has been that there exists a class of physically interesting time evolutions of $u(x,t)$ governed by nonlinear partial differential equations for which the time evolution of $S(t)$ is governed by linear ordinary differential equations. Thus the time evolution is much simpler in the spectral space than in the configuration space. Once the problem has been solved in spectral space the inverse spectral transform can be obtained by well defined procedures, yielding the final solution. As is clear from this discussion this procedure is analogous to Fourier technique for solving certain linear problems. However, the details will not be pursued any further here [9,20,21].

In the next Chapter we will discuss the methods for obtaining the approximate numerical solutions to the KdV equation.

CHAPTER 3

NUMERICAL METHODS APPLICABLE TO THE KdV EQUATION

3.1 Introduction

As mentioned in earlier chapters, the KdV equation is a nonlinear partial differential equation having localized travelling solutions, called solitons. All nonlinear wave equations with soliton solutions are solvable by the powerful tool of Spectral Transform as discussed briefly in Section 2.2. There are many other problems, however, of equal interest for which the Spectral Transform method cannot be applied. Indeed, it is in this area that many of the unsolved problems related to the soliton - like behaviour occur.

Therefore, a useful approach will be to study the evolution of solutions numerically. Numerical methods can be tested on equations for which analytic results and exact solutions are known, and then applied to equations for which analytical results are not known.

3.2 Basic Numerical Methods

As a tool for the study of partial differential equations, numerical analysis offers a confusing variety of techniques. For a specific equation, the question of the

required for a specific range of the independent variables, the limitations of time and storage of space and machine word length. Equally important is the amount of time and effort available for the development of the appropriate software.

Most of the nonlinear evolution equations can be put in the form

$$u_t = L(u), \text{ or} \tag{3.1}$$

$$u_{tt} = L(u) \tag{3.2}$$

where $L(u)$ is some general non-linear differential operator in space variable x.

In order to treat the equation numerically we must replace boundary conditions at infinity by conditions at some finite boundary. To minimize the effect of an infinite boundary, we can take the boundary at large distance from the regions where u is nonconstant or, where it effectively vanishes. At these boundaries we take u and its space derivatives as zero.

Now the first step in solving a differential equation is to discretize it, i.e., the differential equation must be replaced by an approximately finite system of algebraic equations. There are three basic discretization methods:

1. Finite Difference methods
2. Finite element methods
3. The Method of lines.

Here we will discuss the first two only because of their

relevance to the present problem.

3.3 The Finite Difference Methods

The main idea behind the finite difference method for obtaining the solution of a given differential equation is to approximate the derivatives appearing in the equation by a set of values of function at selected number of points called nodes. The most popular way to generate these approximations is through use of Taylor series, but other approaches can be used. One possibility is the direct application of the interpolation formulae.

If $x_i + h = x_{i+1}$, then from Taylor series, we have

$$Df(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \qquad (3.3)$$

where

$$D \equiv \frac{d}{dx} .$$

Another finite difference approximation is the central difference as opposed to the forward difference approximation given by Eq.(3.3). Given below is a central difference approximation to the second order derivative, D^2f:

$$D^2f(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2) \qquad (3.4)$$

Using either the Taylor series or the interpolation formulae, the finite difference approximations to various

derivatives of functions of one or more variables can be constructed. The use of these approximations in the differential equation together with the initial boundary conditions yields an algebraic system of equations which can be solved, using appropriate techniques. Such schemes have been used for the nonlinear KdV equation by Zabusky and Kruskal [8], and Grieg and Morris [14]. Whereas, the results obtained in this work (see Chapter 5) are compared with their work, we will, however, not discuss the details of their schemes.

3.4 Finite Element Methods

Let the partial differential equation be of the form

$$Lu = f \tag{3.5}$$

where L is the differential operator, then the finite element method involves the following steps:

1. Select some known basis functions to approximate, say u(x) in one dimension:

$$u(x) \approx \sum_{j=1}^{N} \alpha_j \varphi_j(x) = U(x,\alpha) \tag{3.6}$$

Here $\varphi_j(x)$ are basis functions, and α denotes the set of α_i's.

2. Substitute $U(x,\alpha)$ into Eq.(3.5); then the unknown coefficients α_i will be approximated by solving the system $LU = f$.

3. Since step 2 is generally impossible to do exactly (one has only n unknowns available to satisfy equations which hold for all values of x in some domain), select N suitable conditions to obtain the N unknowns α_i.

The method outlined above is more general than the finite element method. The finite element method requires, in addition that the basis functions be finite elements; i.e., functions which are zero except on a small part of the domain under consideration. The piecewise polynomials are typical examples of finite element basis functions. The most commonly used basis elements are piecewise linear.

To obtain the unknowns α_i's, one proceeds as follows: The approximate solution $U = \sum_j \alpha_j \varphi_j(x)$ Eq.(3.6) is substituted in Eq.(3.5). Since U is not an exact solution, there results a residual

$$R(x,\alpha) = LU - f \tag{3.7}$$

which is generally not equal to zero. Our aim will be to select the coefficients α_i in such a way that the residual is forced to be zero in some average sense. This can be accomplished by selecting a set of weighting functions w_j's and then setting the integrated weighted residuals to zero. We write it symbolically as

$$(R,\ w_j) = \int_\Omega R(x,\alpha)\ w_j\ d\Omega = 0,\ (j = 1,2,....N) \tag{3.8}$$

Here Ω is the domain of interest.

The solution of Eq.(3.8) will provide us the values for the set of N unknown coefficients. The weighting function w_j can be chosen in many ways. Following approaches are generally adopted.

3.4.1 The Collocation Method

An obvious choice to determine α_i's is to make $LU(x,\alpha) = 0$ at N selected points. Here the weighting functions, w_j's, are taken to be shifted Dirac Delta functions, i.e.,

$$w_j(x) = \delta(x-x_j), \quad j = 1,2,\ldots,N \tag{3.9}$$

This function $\delta(x-x_j)$ has the important property that

$$\int_\Omega f(x)\,\delta(x-x_j)\,dx = f(x_j) \tag{3.10}$$

and the coefficients α_j's are determined from the conditions

$$\int_\Omega R(x,\alpha)\,\delta(x-x_j)\,d\Omega = 0, \quad j=1,2,\ldots,N \tag{3.11}$$

The main disadvantage of the method is that the accuracy obtainable depends strongly on the location of x_j [22,23].

3.4.2 The Least Square Method

In this method the residual itself is taken as the weighting function, i.e.,

$$w_j = R(x,\alpha) . \tag{3.12}$$

Then the coefficients are determined by minimizing the integral

$$I = \int_\Omega R^2(x,\alpha)\, d\Omega = \int_\Omega (LU - f)^2\, d\Omega , \qquad (3.13)$$

that is,

$$\frac{\partial I}{\partial \alpha_j} = 0, \quad j = 1,2,\ldots,N . \qquad (3.14)$$

Here the integral of square of the error is minimized, hence the name, least square method.

3.4.3 The Galerkin Method

Perhaps the best known method based on the weighted residual principle is the Galerkin method. Here the weighting function is chosen to be one of the basis function itself, i.e, $w_j(x) = \varphi_j(x)$, and the constraint imposed for the determination of the coefficients α_j's is:

$$(R(x,\alpha), \varphi_j) = 0$$

or

$$(LU - f, \varphi_j) = 0 \qquad (3.15)$$

Since φ_j's (basis functions) form a complete set, the Galerkin method can be interpreted as forcing all residuals to be orthogonal to the specified basis functions. Therefore good numerical results can be expected and are in fact obtained.

3.5 A Comparison of Different Methods

Now a choice of method will depend mainly on the nature of the problem to be solved, i.e. the order of the differential equation, boundary conditions and continuity requirements. If one has a rectangular domain, then the finite

element methods becomes simpler and more efficient to set up. If one has a more complicated domain, the finite difference method becomes more complicated and less accurate, whereas the finite element methods are relatively unaffected. The principal advantage of the finite element methods is that they are generally more accurate. The second advantage, as indicated above, is that they are more flexible in handling complex geometrical shapes by the use of arbitrary shaped simple elements.

3.6 Convergence, Consistency and Stability

Whatever approach is chosen, one would ideally like to show that the method is convergent [24], i.e. the global error for some fixed and finite value of time tends to zero as the step length (nodal separations) tend to zero, or as the number of basis functions tend to infinity. Unfortunately, the proof of convergence is usually extremely difficult, even when possible, and a satisfactory theory is only available in general for linear equations. Most discussions of nonlinear methods are based on an analysis of the equation linearized about some constant solution.

To ensure convergence, methods in general must be consistent and stable. A consistent method is one in which the truncation error tends to zero as the step lengths in the problem tend to zero. Sometimes this limit must be taken in a specific way; for example in the Hopscotch method the limit

must be such that $k/h \to 0$ as $k, h \to 0$, where h and k are step lengths in space and time respectively. Stability means that some norm of the approximate solution must remain bounded as $n \to \infty$, where $nk = t$ for a fixed t. Even stability is usually difficult to prove for nonlinear equations, and the usual approach is to study the stability of a linearized version of the equation [16,24].

In the next chapter, we discuss the development of a finite element computer code for the KdV equation using cubic splines [27,28] . As will be seen there, the use of finite elements in space for the KdV equation yields a nonlinear system of ordinary differential equations which are solved by a finite difference method. The results obtained will be presented and discussed in latter chapters.

CHAPTER 4

NUMERICAL COMPUTATIONS AND RESULTS

4.1 Preliminary Remarks

In our work we have used Finite Element Method (FEM), and in particular the Galerkin method using cubic splines for the solution of the KdV equation. The KdV equation is generally written in the form:

$$K[u] \equiv u_t + uu_x + \varepsilon u_{xxx} = 0 \qquad (4.1)$$

where ε is a positive constant (see Section 1.2). As mentioned in earlier chapters it possesses the single-soliton solution:

$$u(x,t) = 3c \ \mathrm{Sech}^2 (kx - wt + d) \qquad (4.2)$$

where

$$k = \frac{1}{2}\left(\frac{c}{\varepsilon}\right)^{1/2}, \quad \text{and}$$

$$w = kc \qquad (4.3)$$

Here c and d are parameters, c is the velocity of the soliton, and d is the phase or central shift.

4.2 Calculation of Galerkin Constants

We take the interval of study as [0,2] on the x-axis, because at the boundaries of this interval the solution almost vanishes for the values of c, and d for which we will be

interested and over the time range of interest.

Now the interval [0,2] of the x-axis is divided into N subintervals or elements each of which has a length $h = \frac{2}{N}$. We approximate the solution as

$$U = \sum_{j=2}^{N-2} \alpha_j(t)\, \varphi_j(x) \tag{4.4}$$

where the basis functions $\varphi_j(x)$ are approximately shifted basic cubic splines. These will have the local support on the interval $[(j-2)h, (j+2)h]$.

Now substituting (4.4) into (4.1), we get the residual

$$R(x,\alpha) = U_t + UU_x + \varepsilon U_{xxx} \tag{4.5}$$

As mentioned in Chapter 3, in the Galerkin method, we take the weighting functions also as $\varphi_j(x)$ and impose the condition:

$$\int_a^2 R(x,\alpha)\, \varphi_j(x)\, dx = 0, \quad j=2,3,\ldots,N-2 \tag{4.6}$$

where the detailed expressions for $R(x,\alpha)$ is seen to be

$$R(x,\alpha) \equiv \frac{\partial}{\partial t}\left(\sum_j \alpha_j(t).\varphi_j(x)\right) + \sum_j \alpha_j(t)\, \varphi_j(x). \frac{\partial}{\partial x}\left(\sum_j \alpha_j(t).\varphi_j(x)\right) + \frac{\partial^3}{\partial x^3}\left(\sum_j \alpha_j(t),\varphi_j(x)\right) \tag{4.7}$$

where the summation over j is from 2 to (N-2) as in Eqs.(4.4) and (4.6).

Substituting Eq.(4.7) into Eq.(4.6) we get the following system of coupled ordinary differential equations for the time-dependent expansion coefficients, $\alpha_j(t)$

$$A_{ij}\ \frac{d\alpha_j}{dt} + h^{-1}(B_{ijk})\ \alpha_j\alpha_k - h^{-3}(C_{ij})\alpha_j = 0 \qquad (4.8)$$

Here the summation over the repeated indices (i.e. if they are repeated in the same term) is implied from 2 to (N-2). The Galerkin constants A_{ij}, B_{ijk} and C_{ij} are given by

$$A_{ij} = h^{-1}(\varphi_i,\varphi_j) \equiv h^{-1} \int_{-\infty}^{+\infty} \varphi_i(x).\varphi_j(x)\ dx$$

$$B_{ijk} = (\varphi_i, \varphi_j\varphi_k^{(1)}) \equiv \int_{-\infty}^{+\infty} \varphi_i(x).\varphi_j(x)\ \varphi_k^{(1)}(x)\ dx$$

$$C_{ij} = h^2(\varphi_i^{(1)},\varphi_j^{(2)}) \equiv h^2 \int_{-\infty}^{+\infty} \varphi_i(x)\ \varphi_j^{(3)}(x)dx$$

$$= h^2 \int_{-\infty}^{+\infty} \varphi_i^{(1)}(x)\varphi_j^{(2)}(x)\ dx \qquad (4.9)$$

Since our region of interest is $[0,2]$, the above integrations are done in the interval $[0,2]$. Furthermore, the basic cubic spline functions $\varphi_j(x)$ are spread over a compact local support of 4h only. Hence the integration exists over an interval of 4h giving the band nature of the matrix.

As mentioned in the beginning, the basis functions φ_j's have the support over the interval $[(j-2)h, (j+2)h]$. Therefore, φ_1 will be spread over the interval $[-h, 3h]$. Since we have taken our interval as $[0,2]$, we take φ_2 as the first basis function. Similarly our last basis function will be φ_{N-2} and, not φ_{N-1}, because the latter will extend to one step beyond the interval.

Now, we write a basic cubic spline in the following standard form [16,28]

$$
\begin{aligned}
M(y) &= 0 && y \leq -2 \\
&= \tfrac{1}{6}[(y+2)^3] && -2 \leq y \leq -1 \\
&= \tfrac{1}{6}[(y+2)^3 - 4(y+1)^3] && -1 \leq y \leq 0 \\
&= \tfrac{1}{6}[(-y+2)^3 - 4(-y+1)^3] && 0 \leq y \leq 1 \\
&= \tfrac{1}{6}[(-y+2)^3] && 1 \leq y \leq 2 \\
&= 0 && 2 \leq y
\end{aligned}
\qquad (4.10)
$$

Then the basis functions $\varphi_j(x)$ can be written as

$$\varphi_j(x) = M\left(\frac{x}{h} - j\right) \qquad (4.11)$$

Once the basis functions are known we can integrate (4.9) to get the Galerkin constants.

In calculating these constants, the first thing. We will need is a subroutine to multiply two polynomials because of the product term present in the integrand. The product polynomial can then be integrated easily by suitably changing the power and the coefficient of each term automatically

by the computer code. The integration can further be simplified by changing the limits of integrations to [0,1] in all cases by suitable rescaling of the variable of integration. Since the polynomials are defined piecewise on intervals of h, the integrals can always be reduced to the form:

$$\int_0^1 [a_0 + a_1 y + \ldots\ldots a_p y^p]\, dy = \sum_{k=0}^{p} a_k/(k+1) \tag{4.12}$$

4.3 Initial Conditions

After the calculation of the Galerkin constants, we will need a subroutine to solve the system of coupled nonlinear ordinary differential equations, (4.8). Hence the initial value of the dependent variables, $\alpha_j(0)$, need to be specified. Here we will use least square technique to calculate the initial values. We minimize the expression

$$E \equiv \| \sum_j \alpha_j(0)\, \varphi_j(x) - f_o(x) \|^2_{L_2}$$

and

$$= \left(\sum_j \alpha_j(0)\, \varphi_j(x) - f_o,\ \sum_k \alpha_k(0) \varphi_k(x) - f_o \right) \tag{4.13}$$

where $f_o \equiv u(x,0)$, and $u(x,t)$ is given by Eq.(4.2). The particular numerical values of , c and d needed to evaluate $u(x,0)$ are given below.

Setting $\frac{\partial E}{\partial \alpha_j(0)} = 0$, we obtain the following equations from which to compute $\alpha_j(0)$:

$$\sum_j A_{ij}\alpha_j(0) = (f_o, \varphi_i)/h \qquad (4.14)$$

where A_{ij} can be seen to be the same as given by the first of Eq.(4.9).

Since the integral on the right hand side of Eq.(4.14) cannot be easily evaluated analytically, we have used a numerical integration to perform the required integration.

Here the matrix formed by A_{ij} is a band symmetric matrix of order (N-3). We can save much space in storage by utilizing its band nature, i.e. storing it in band symmetric storage mode. The number of co-diagonals of the symmetric matrix formed by A_{ij} is 3 here.

4.4 Solution of the ODE System

The Eqs.(4.8) may be written in the form

$$A_{ij} \frac{d\alpha_j}{dt} = h^{-3} C_{ij}\alpha_j - h^{-1} B_{ijk\ j\ k} \qquad (4.15)$$

which is equivalent to the following matrix form:

$$A \frac{D\alpha}{DT} = F(\alpha, T) \qquad (4.16)$$

where A is the band symmetric matrix and $F(\alpha, T)$ is a column vector. In the present case, however, F does not depend explicitly on time. Since available differential equation solvers [IMSL, NAG Libraries], generally requires the system of equations in the form $\frac{D\alpha}{DT} = F(\alpha, T)$, we need to convert

Eq. (4.15) in the above form by calculating A^{-1} and then writing in the desired form:

$$\frac{D\alpha}{DT} = A^{-1} \quad F(\alpha,T) \qquad (4.17)$$

The system of Eqs.(4.17) is then solved using standard subroutines LIN2PB and DREBS from IMSL library.

4.5 Evaluation of the Cubic Spline for the Final Solution

Once the $\alpha_j(t)$, $j = 2,\ldots,N-2$ have been obtained at the desired times T by solving the system of equations (4.17), the values of U(X,T) can be evaluated using Eqs.(4.4) and (4.11).

Figure (4.1) gives an overall outline of the computer code developed for carrying out the above-mentioned tasks. The details of the subroutines used are given in Appendix B, and the results of some sample calculations are presented in Appendix C. In the following section, we give a summary of these calculations, particularly with a view to bring out the accuracy of the computed solutions.

4.6 Results and Their Accuracy

The actual numerical computations were done for the following values of the parameters in Eq. (4.2) :

$$\begin{aligned} \varepsilon &= 4.84\times10^{-4} \\ c &= 0.3 \\ d &= -k = \tfrac{1}{2}\left(\tfrac{c}{\varepsilon}\right)^{1/2}, \text{ and} \\ w &= kc \ . \end{aligned} \tag{4.18}$$

These values were chosen to facilitate comparison with previous works. For the same reasons the interval of interest was taken as $[0,2]$. It can be seen from the actual initial data as well as its cubic spline representation given in Table (4.1), that at the ends of the interval the initial values are negligibly small and therefore can be taken as zero. Furthermore we will restrict the evolution of the solution to a maximum of T = 2.0 second. Upto this time, it can be easily seen that the single-soliton solution continues to remain negligibly small at the end points.

We have calculated the solution for two different grids, N = 40 and N = 50. In the former case the final solution was computed for T = 0.5, 1.0, 1.5 and 2.0 seconds. For N = 50 the solution was obtained for T = 0.4, 0.8 and 1.2 second. The results are summarized in Tables (4.1) to (4.6), and Figs. (4.2) and (4.3).

Table (4.1) shows at alternative grid points the values of the cubic spline approximation of the initial data for N = 40. It is seen that an maximum error of approximately 0.4 percent is incurred when compared to the peak value. Therefore, one perhaps cannot expect an accuracy better than

this in the final solutions. (This is confirmed later). The corresponding error for N = 50 is seen to be about 0.2 percent.

In Fig. (4.2) the initial data at T = 0.0, the calculated and expected solutions for T = 2.0 are plotted. The figure clearly brings out the overall soundness of the computer code developed. It is infact difficult to distinguish between the calculated and expected solutions graphically. The same is true for Fig. (4.3) where the expected and calculated solutions are shown for T = 1.2 using N = 50.

Tables (4.2) and (4.3) show the calculated and expected positions of the single-soliton peaks at different times for N = 40 and 50 respectively. It is seen that in all cases the peaks are observed precisely where expected.

In Table (4.4) a detailed comparison of the calculated and expected solutions, and the corresponding absolute and relative errors at alternative grid points, are given. The maximum error expressed as a percentage of the peak value is seen to be approximately 2 percent. Here it may be recalled that the corresponding error in the initial data representation was about 0.2 percent (see Table 4.1 and the discussion above). The L_∞ and L_2 errors are also given at the bottom of Table (4.4).

Tables (4.5) and (4.6) give the errors in the peak values only for various runs. It is seen that the error

is generally less for N = 50 than it is for N = 40. Similarly it generally increases if time of evolution is increased. It should be pointed out, however, that it is L_∞ and L_2 errors which give a more systematic idea of the accuracy of calculations. These will be presented in Chapter 5, where a detailed comparison will also be made with previous works.

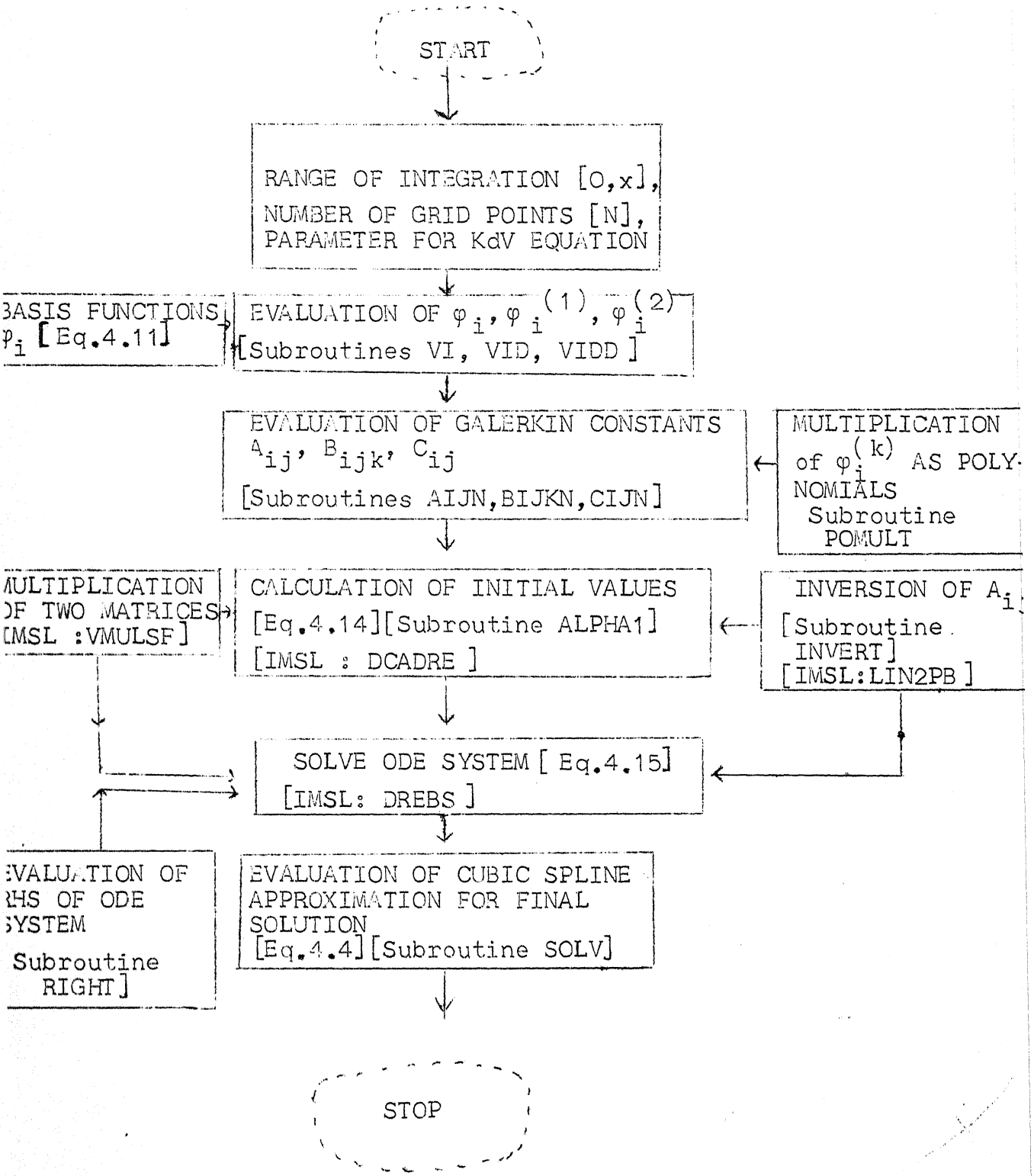

FIG. 4.1 AN OUTLINE OF THE PRESENT COMPUTER CODE

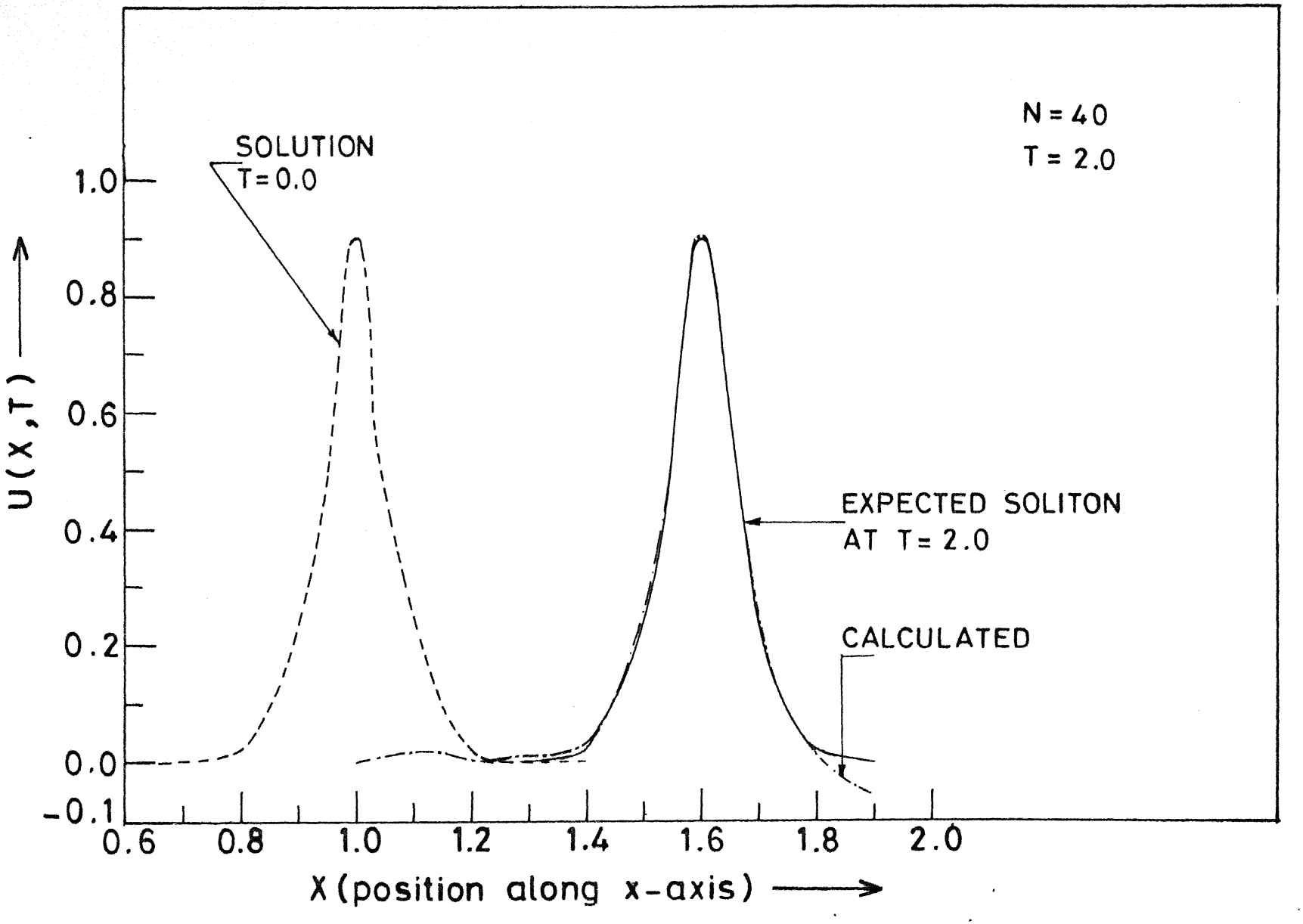

FIG. 4.2 INITIAL AND CALCULATED/EXPACTED POSITIONS OF KdV

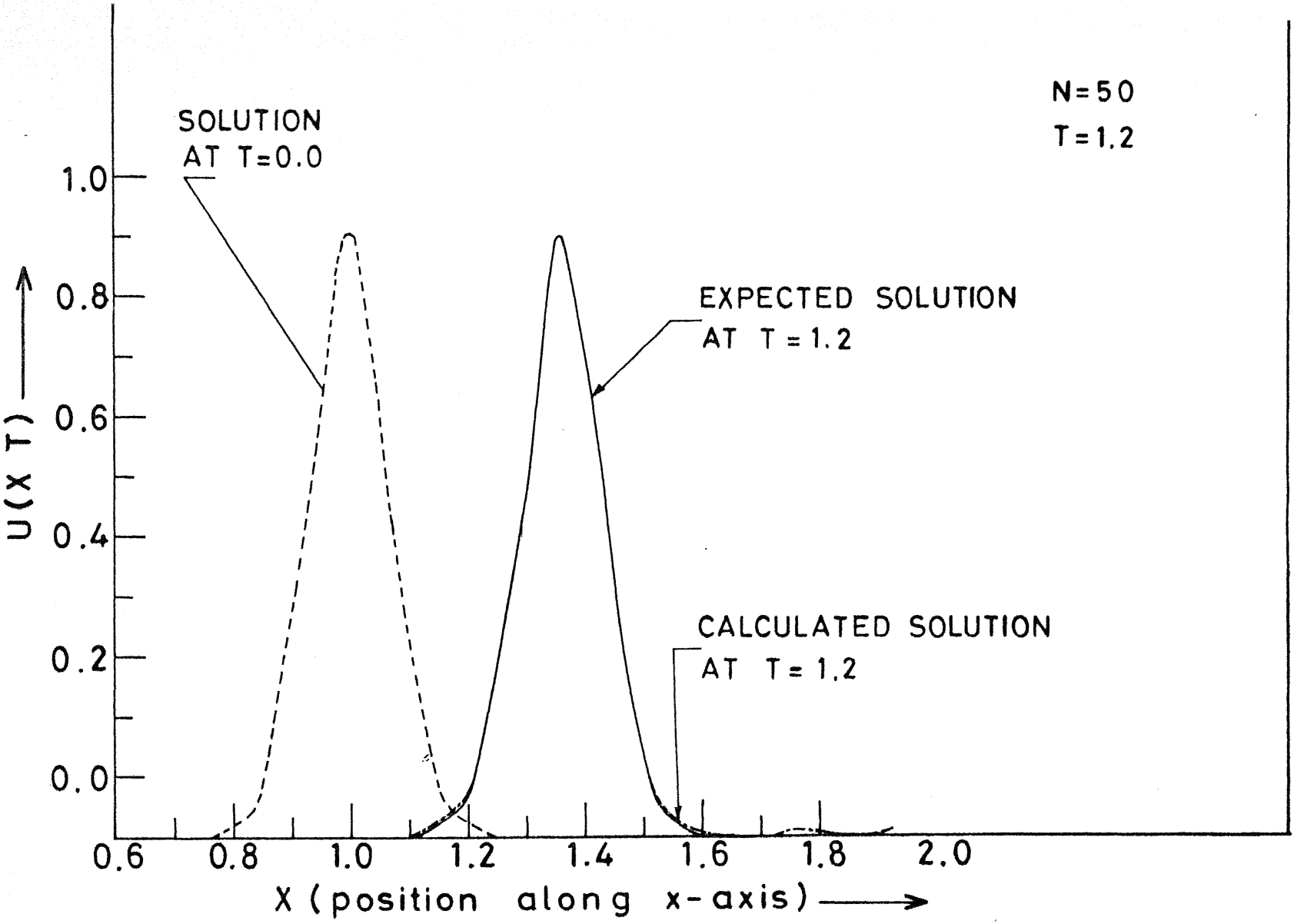

FIG. 4.3 INITIAL AND CALCULATED/EXPECTED POSITIONS OF KdV SOLITON AT T=1.2 SECOND.

TABLE 4.1

Comparison of Actual Initial Data and Its Cubic Spline Representation (N = 40)*

S. No.	X	Actual Initial Data $U_o(X)$	Cubic Spline Representation $U_o(X)$
1	0.20	0.80608E-08	-0.23813E-06
2	0.40	0.11718E-05	-0.19135E-05
3	0.60	0.17033E-03	0.13309E-03
4	0.80	0.24427E-01	0.24024E-01
5	1.00	0.90000E+00	0.90392E+00
6	1.20	0.24427E-01	0.24024E-01
7	1.40	0.17033E-03	0.13309E-03
8	1.60	0.11718E-05	-0.19135E-05
9	1.80	0.80608E-08	-0.23808E-06

*

Notes: 1. L_∞ error (N=40) = 3.92E-3, L_2 error (N=40) =1.77E-3

2. L_∞ and L_2 errors in N=50 representation were found to be 2.04E-3 and 0.81E-3 respectively.

TABLE 4.2

Errors in the Observed Locations of the Soliton Peaks
(N = 40)

S.No.	T (second)	X^*_{peak}		Error in Peak Position
		Calculated Position	Expected Position	
1	0.0	1.00	1.00	0.0
2	0.5	1.15	1.15	0.0
3	1.0	1.30	1.30	0.0
4	1.5	1.45	1.45	0.0
5	2.0	1.60	1.60	0.0

*X_{peak} denotes the position of the peak of KdV soliton along the x-axis.

TABLE 4.3

Errors in the Observed Locations of the Soliton Peaks
(N = 50)

S.No.	T (second)	X^*_{peak}		Error in Peak Position
		Calculated Position	Expected Position	
1	0.0	1.00	1.00	0.0
2	0.4	1.12	1.12	0.0
3	0.8	1.24	1.24	0.0
4	1.2	1.60	1.60	0.0

*X_{peak} denotes the position of the peak of KdV soliton along the x-axis.

TABLE 4.4

A Detailed Comparison of the Calculated and Expected Single-Soliton Solutions*
(T = 1.2 second, N = 50)

S.No.	X	Calculated Value of U(X,T)	Expected Value of U(X,T)	Absolute Error ($\times 10^3$)	Percent Error (Relative to Peak Value
1	0.48	0.60520E-02	0.11000E-08	6.05	0.67
2	0.56	0.10978E-01	0.80508E-08	10.00	1.11
3	0.64	0.56327E-02	0.59071E-07	5.63	0.62
4	0.72	-0.81609E-02	0.43288E-06	8.16	0.90
5	0.80	0.49477E-02	0.31722E-05	4.94	0.54
6	0.88	-0.82831E-02	0.23246E-04	8.25	0.91
7	0.96	0.65387E-02	0.17033E-03	6.36	0.70
8	1.04	-0.54447E-02	0.12475E-02	6.69	0.74
9	1.12	0.10860E-01	0.91017E-02	1.75	0.19
10	1.20	0.64241E-01	0.64509E-01	0.36	0.40
11	1.28	0.37732E+00	0.38037E+00	3.05	0.33
12	1.36	0.90181E+00	0.90000E+00	1.81	0.20
13	1.44	0.39094E+00	0.38037E+00	10.57	0.11
14	1.52	0.61542E-01	0.64509E-01	3.06	0.34
15	1.60	0.17014E-01	0.91017E-02	7.91	0.88
16	1.68	0.58802E-03	0.12475E-02	0.65	0.07
17	1.76	0.14149E-01	0.17033E-03	13.97	1.55
18	1.84	-0.24656E-02	0.23246E-04	2.48	0.28
19	1.92	0.17256E-01	0.31722E-05	17.25	1.91

*The L_∞ and L_2 errors for the these calculations are 17.25E-3 and 8.70E-3 respectively. These errors for this and other cases are summarized and compared with available literature in Tables 5.1 to 5.4.

TABLE 4.5

Errors in Peak Values for Different Evolution Times (T)
(N = 40)*

S.No.	T	Calculated Value	Expected Value	Absolute Error in Peak ($\times 10^3$)	Relative Error (percent)
1	0.5	0.91550	0.90000	15.50	1.72
2	1.0	0.91056	0.90000	10.56	1.11
3	1.5	0.88619	0.90000	13.81	1.53
4	2.0	0.91593	0.90000	15.93	1.77

* Here we have tabulated the errors in peak values only. The L_∞ as well as L_2 errors for the complete solutions can be found in Tables 5.1 to 5.4.

TABLE 4.6

Errors in Peak Values for Different Evolution Times (T)
(N = 50)*

S.No.	T (second)	Calculated Value	Expected Value	Absolute Error in Peak ($\times 10^3$)	Relative Error (percent)
1	0.4	0.90582	0.90000	5.82	0.64
2	0.8	0.91128	0.90000	11.28	1.25
3	1.2	0.90181	0.90000	1.81	0.20

* Here we have tabulated the errors in peak values only. The L_∞ as well as L_2 errors for the complete solutions can be found in Tables 5.1 to 5.4.

CHAPTER 5

COMPARISON WITH PREVIOUS WORKS AND CONCLUSIONS

5.1 Comparison of L_∞ and L_2 Errors

A variety of finite element schemes and some finite difference schemes have been used for the numerical solution of KdV equation by previous workers. Here we compare our results with some of the previously available results.

Alexander and Morris [16] have used a Galerkin scheme using cubic splines for basis functions and an appropriate set of quintic boundary functions. They have also used a dissipative Galerkin scheme and presented their results for different values of dissipation parameter, q. However, they have evolved the single-soliton solution upto 0.4732 second, whereas, we have evolved the solution upto 2 seconds. Furthermore, they have given only L_∞ errors in their calculations. In Table (5.1) we compare our results with the above-mentioned calculations. We find that the L_∞ error in our run for T = 0.5 is less than the error for q = 0 and T = 0.3958 in [16] . The errors in other runs also compare somewhat favorably for the present work.

In Table (5.2) we compare our calculations with N = 50 with those of Alexander-Morris [16] with N = 60. Here the results of the present work appear to be appreciably

better as can be seen from the table. In particular when we evolve the solution to T = 1.2, an L_{∞}- errors of 17.3×10^{-3} is incurred which is almost comparable to the L_{∞}-error in [16], when the solution is evolved only to 0.4617 second.

In Table (5.3), the errors are compared with the finite difference schemes of Zabusky-Kruskal [8] and Grieg-Morris [14]. Here both the L_{∞} and L_2 errors in the present work are found to be a factor of 2 to 10 times less in the present work.

Lastly in Table (5.4) we compare our results with some recently used Petrov-Galerkin schemes [17]. Here we find that the L_2 - errors in the present calculations are about a factor of 4 less than those in the Petrov-Galerkin schemes. However, one of the later schemes appears to have somewhat less L_{∞}-errors. We also present a rough comparison of computation times in Table (5.5).

5.2 Conclusions and Suggestions

In conclusion we may say that the present attempt to develop a computer code for the numerical evolution of nonlinear wave equations has been highly successful and the results obtained already compare quite favorably with the previous works. However the present code needs to be developed and generalized in many directions, or a fresh more general code can be written based on the experience obtained

here, we list some of the points below:

1. The exponential, trigonometric or hyperbolic splines may be used. An exponential/hyperbolic polynomial may be particularly useful in the present context as a boundary function.

2. A variable/adjustable grid can be used.

3. The code should be written such that it can handle a variety of basis and weighting functions, as well as assemble the Galerkin constants for many nonlinear evolution equations of interest.

4. The finite element method can be used for time variable also.

TABLE 5.1

Comparison of Errors with Previous Works

I. Comparison with Alexander-Morris Galerkin Schemes [16] (N = 40)

S.No.	T (second)	Alexander-Morris Galerkin Schemes (Error x 10^3) L_∞				Present Galerkin (Error x 10^3)		
		q = 0*	q = 1.0	q = 20.661	q = 103.31	T (second)	L_∞	
1	0.3605	–	–	59.0	–	0.5	37.0	18
2	0.3958	57.0	–	–	–	1.0	52.7	24
3	0.4155	–	27.0	–	–	1.5	50.7	30
4	0.4732	–	–	–	25.0	2.0	53.6	31

*q here denotes the dissipation parameter in the dissipation Galerkin scheme.

TABLE 5.2

Comparison of Error with Previous Works

II. Comparison with Alexander-Morris Galerkin Schemes (N = 50 or 60)

S.No.	Alexander-Morris Galerkin Schemes (Error x 10^3)(N = 60)			Present Galerkin-Scheme (Error x 10^3)(N=50)		
	T (second)	L_∞		T (second)	L_∞	L_2
		q = 0*	q = 1			
1	-	-	-	0.4	10.5	5.3
2	0.4617	15.3	18.0	0.8	11.4	7.5
3	-	-	-	1.2	17.3	8.70

* q here denotes the dissipation parameter in the dissipative Galerkin scheme.

TABLE 5.3

Comparison of Error with Previous Works

III. Comparison with Finite-Difference Schemes (N = 40)

S. No.	T (second)	Error x 10^3 in Zabusky-Kruskal (Leap-Frog Scheme)		Greig-Morris (Hopscotch Scheme)		The Present Method (Galerkin Scheme)	
		L_∞	L_2	L_∞	L_2	L_∞	L_2
1	0.25	19.4	34.64	32.7	61.21	–	–
2	0.50	63.5	122.68	67.4	122.41	37.0	18.1
3	0.75	122.4	210.44	99.3	181.35	–	–
4	1.00	161.4	298.19	141.6	228.10	52.7	24.7
5	1.50	–	–	–	–	50.7	30.35
6	2.00	–	–	–	–	53.6	31.60

TABLE 5.4

Comparison of Error with Previous Works

IV. Comparison with Petrov-Galerkin Schemes

S. No.	T (second)	Error x 10^3 in					
		Petrov-Galerkin Scheme		Modified Petrov-Galerkin Scheme		The Present Meth (Galerkin Scheme)	
		L_∞	L_2	L_∞	L_2	L_∞	L_2
1	0.25	42.18	81.39	32.22	52.15	–	–
2	0.50	51.85	102.54	22.85	64.90	37.0	18.1
3	0.75	87.60	125.84	35.86	89.01	–	–
4	1.00	100.41	150.57	39.39	107.20	52.7	24.7
5	1.50	–	–	–	–	50.7	30.35
6	2.00	–	–	–	–	53.6	31.60

TABLE 5.5

Computation Time for Different Runs*

S.No.	N = 40		N = 50	
	Evolution Time (T) (second)	Computation Time (second)	Evolution Time (T) (second)	Computation Time (second)
1	- 0.5	36.72	0.4	43.73
2	1.0	70.79	0.8	89.22
3	1.5	103.81	1.2	128.33
4	2.0	120.16	-	-

* These may be compared with the times quoted in the range of approximately 15 to 65 seconds time/1 real second in Alexander-Morris [16] for N = 40 and 60. The present calculations were done on DEC 1090 and no effort was made to optimize on the computation time.

APPENDIX A

TWO TRANSFORMATIONS FOR EQ. (1.1) TO EQ. (1.4)

By trying the transformations

$$u \rightarrow \alpha u,$$

$$t \rightarrow \beta t, \text{ and}$$

$$x \rightarrow \gamma x,$$

where α, β and γ are constants, one finds that the following values of α, β and γ transform Eq.(1.1) to Eq.(1.4):

I. $\alpha = B^{-1} C^{1/3}$

$\beta = A$

$\gamma = C^{1/3}$

II. $\alpha = B^{-\frac{3}{5}} C^{\frac{1}{5}}$

$\beta = A B^{-\frac{3}{5}} C^{\frac{1}{5}}$

$\gamma = B^{-\frac{1}{5}} C^{\frac{2}{5}}$.

APPENDIX B

LISTING OF THE COMPUTER CODE

```
@@@
```

```
VALUES, VCVTFO, LIN2PB & DREBS IN SUBSIX.FOR,
RIGHT IN SUBONE.FOR
SOLV IN SUBFIVE.FOR.
-----------------------------------------------------------------------
IF BSSM SUBROUTINE IS CALLED IT WILL USE ANOTHER SYSTEM
SUBROUTINE VCVTFO WHICH STORES A FULL BAND SYMMETRIC MATRIX IN
SYMMETRIC BAND STORAGE MODE BUT IT IS PREFEREBLE TO READ THE
VALUES IN BAND SYMMETRIC STORAGE MODE BECAUSE OF STORAGE FACTORSRS.
THE DETAILS OF SUBTOUTINE BSSM IS GIVEN IN PROGRAM SUBTHREE.FOR.
-----------------------------------------------------------------------
THE DETAILS OF VARIBLES USED IS GIVEN BELOW:
A(I,J) IS A 100 BY 4 MATRIX WHICH CONTAINS THE VALUES OF
COEFFICIENT A IN BAND SYMMETRIC STORAGE MODE.
C IS A 100 BY 1 MATRIX WHICH HAS BEEN USED TO STORE THE
MULTIPLICATION OF TWO MATRICES AINV AND BS WHEN CALCULATING THE
INITIAL VALUES AND MULTIPLICATION OF AINV AND BR WHEN CALCULATING
FINAL VALUES.
AINV IS A VECTOR WHICH CONTAINS THE INVERSE OF THE BAND SYMMETRIC
MATRIX A STORED IN BAND SYMMETRIC STORAGE MODE.
BR CONTAINS THE RIGHT HAND SIDE OF THE EQUATION OF THE FORM
A DY/DT=B.
VARIABLE Y IN THE STARTING CONTAINS THE INITIAL VALUES OF DEPE
DEPENDENT VARIABLE AND FINALLY THE VALUES OF SOLUTION AT T=TEND.
B(I,J,K) IS A THREE DIMENSIONAL MATRIX WHICH CONTAINS THE VALUES
OF GALERKIN CONSTANT (VALUES CALCULATED BY BIJKN).
CC IS ALSO THE GALERKIN CONSTANT(VALUES CALCULATED BY CIJN).
ALPA IS A VECTOR WHICH TRANSFERS THE VALUES OF Y TO SUBROUTINE
SOLV.
UXT IS A VECTOR WHICH CONTAINS FINAL SOLUTION,i.e
U(x,t)=SUM ALPA(J)(t)*Y(J)(x)
EXTERNAL ROUTINE CALCULATES THE VALUES OF DY/DT=CONSTANT AND
TRANSFERS IT TO DREBS WHICH FINALLY GIVES Y(t).
NI STANDS FOR NUMBER OF TOTAL GRID POINTSAND NDIM IS THE
ROW DIMENSIONS OF A,C,BS,BR,Y,R,S,ALPA & UXT AS DECLEARED IN
THE MAIN PROGRAM.
-----------------------------------------------------------------------
REAL A(100,4),C(100,1),BS(100,1),AINV(5050),BR(100)
REAL Y(100),R(100),S(100),WK(2900)
REAL B(5,5,5),CC(5,5),ALPA(100),UXT(100)
-----------------------------------------------------------------------
EXTERNAL DFN
COMMON AINV,N,NC,M,IB,IC,B,CC,HC,EPS
OPEN(UNIT=1,DEVICE='DSK',FILE='BSSM.DAT')
OPEN(UNIT=2,DEVICE='DSK',FILE='COMP.DAT')
ACCEPT*,NI
N=NI-2
NDIM=100
NM=N-1
NC=3
IA=NDIM
IAA=NDIM
```

```
IP=[illegible]
IT=[illegible]
M=1
X0(1)=0.0
X0(2)=.1
EPS=1.0E-04
AER=.0001
RER=.0001
C-----------------------------------------------------------------------
FOLLOWING CONSTANTS ARE TO BE USED BY THE ROUTINE DREBS.
H IS STEP SIZE IN TIME(DETAILS ARE GIVEN IN SUBROUTINE
SUBSIA.FOR).
JSTART=0
JM=2
H=.01
T=0.0
HMIN=.001
ACCEPT*,TEND
TOL=.02
IVO=2
C-----------------------------------------------------------------------
HERE START STANDS FOR INITIAL POINT AND FINAL FOR END POINT OF
RANGE OF INTEREST,THEREFORE HC IS STEP SIZE IN SPACE.
START=0.0
FINAL=2.0
NB=5
NT=5
HC=(FINAL-START)/FLOAT(NT)
C-----------------------------------------------------------------------
READ(1,*)((A(I,J),J=1,4),I=1,NM)
CALL BSSM(NM,NC,N,IAA,HC,A)
CALL BIJKN(NB,HC,B)
CALL CIJN(NC,HC,CC)
CALL ALPHA1(N,CONT1,HC,EPS,AER,RER,BR)
CALL ALPHA2(N,CONT1,CONT2,HC,EPS,AER,RER,BR)
DO 1 I=2,N
BS(I-1,1)=BR(I)
CONTINUE
CALL INVERT(NM,NDIM,NC,A,AINV)
CALL VMULSF(AINV,NM,BS,M,IB,IC,ICD)
DO 2 I=1,NM
Y(I)=X(I,1)
S(I)=X(I,1)
ALPA(I+1)=X(I,1)
CONTINUE
CALL SOLN(N,HC,ALPA,UXT)
WRITE(2,33)
WRITE(2,34)
FORMAT(10X,'FOLLOWING ARE THE SOLUTIONS U(x,t)'/)
FORMAT(1X,'-----------------------------------
1-----'/)
```

```
WRITE(2,33) T,CONT1,HC,H,NT
FORMAT(1X,' SOLUTIONS AT T=',F5.3//1X,'C=',F5.2//1X,'STEP SIZE
1 IN SPACE HC=',F5.2//1X,'STEP SIZE IN TIME H=',F5.2//1X'N=',I3//
21X,'----------------------------------------------------------
3----------')
WRITE(2,11) (UXT(I),I=2,N)
FORMAT(1X,5(1X,E12.5)/)
CONTINUE
H=AMIN1(H,TEND-T)
CALL DREBS(DFN,Y,T,NM,JM,IND,JSTART,H,HMIN,TOL,R,S,WK,IER)
IF(IER.NE.0) GO TO 15
IF(T.LT.TEND-HMIN) GO TO 5
CONTINUE
DO 22 I=1,NM
ALPA(I+1)=Y(I)
CONTINUE
CALL SOLV(N,HC,ALPA,UXT)
WRITE(2,44)
FORMAT(1X,'----------------------------------------------------
1----------')

WRITE(2,10) T,CONT1,HC,H,NT
FORMAT(1X,' SOLUTIONS AT T=',F5.2//1X,'C=',F5.2//1X,'STEP SIZE
1 IN SPACE HC=',F5.2//1X,'STEP SIZE IN TIME H=',F5.2//1X,'N=',I3
2//1X,'----------------------------------------------------------
3----------')
WRITE(2,11) (UXT(I),I=2,N)
FORMAT(1X,5(1X,E12.5)/)
WRITE(2,44)
STOP
CALL JERTST
END
C----------------------------------------------------------------------
NOW FOLLOWS THE DFN SUBROUTINE.THIS ROUTINE CALLCULATES
DY(I)/DT=F(I),I=1,...,NM.
SUBROUTINE DFN(Y,T,NM,DY)
REAL Y(100),DY(100),AINV(5050),BS(100,1),C(100,1),BR(100)
REAL B(5,5,5),CC(5,5)
COMMON AINV,N,NC,M,IB,IC,B,CC,HC,EPS
CALL RIGHT(N,EPS,HC,B,CC,Y,BR)
NM=N-1
DO 100 I=2,N
BS(I-1,1)=BR(I)
CONTINUE
CALL VMULSF(AINV,NM,BS,M,IB,C,ICD
DO 121 I=1,NM
DY(I)=C(I,1)
CONTINUE
RETURN
END
```

```
@@@
```

```
C
C                S U B T W O . F O R
C
C-----------------------------------------------------------------------
C THIS PROGRAM CONTAINS SUBROUTINE AIJN,BIJKN,CIJN FOR THE
C CALCULATION OF GALERKIN CONSTANTS A(I,J),B(I,J,K) & C(I,J).
C THESE SUBROUTINES USE OTHER SUBROUTINES VI,VID,VIDD & POMULT.
C DETAILS OF THESE SUBROUTINES ARE GIVEN AT PROPER PLACES.
C VI SUBROUTINE CALCULATES THE VALUES OF COEFFICIENTS OF THE
C POLYNOMIAL ARISING FROM THE BASIS FUNCTION FOR DIFFERENT VALUES
C OF X. SIMILARLY VID & VIDD  WHEN THE BASIS FUNCTIONS ARE DIFFER-
C ENTIATED ONCE AND TWICE RESPECTIVELY.POMUL SUBROUTINE MULTIPLIES
C TWO POLYNOMIALS OF DIFFERENT ORDERS.
C-----------------------------------------------------------------------
C A IS 39*39 COEFFICIENT MATRIX AND P1&P2 ARE COEFFICIENTS
C OF TWO POLYNOMIALS STARTING FROM CONSTANT TERM
C MP'S ARE COEFFICIENTS OF MULTIPLIED POLYNOMIALS
C LL DENOTES LOWER LIMIT IN EACH INTEGRATION
C-----------------------------------------------------------------------
C                S U B R O U T I N  A I J N                           I
C-----------------------------------------------------------------------
      SUBROUTINE AIJN(NM,H,A)
      DIMENSION A(38,38),P1(10),P2(10)
      REAL MP(20),LL,LO
      CINTEG=0.0;X=0.0
C-----------------------------------------------------------------------
      DO 1000 I=2,NM
      DO 2000 J=2,NM
C SINCE COEFFICIENTS ARE ZERO FOR THE CROSS TERMS WITHT I,J
C INDEXES DIFFERING BY FOUR OR GREATER WE CAN BY PASS THOSE
C TERMS TO SAVE COMPUTATION TIME.
C-----------------------------------------------------------------------
      IJ=IABS(I-J)
      IF(IJ.GT.3) GO TO 50
      CONTINUE
      Y=X/H-FLOAT(I)
      IF(Y.GT.-1.99) GO TO 30
      IF(Y.GT.-2.01) GO TO 50
      LL=(X-H)/H
      LO=FLOAT(I)-LL
      CALL VI(Y,2,P1,N)
      GO TO 500
      CONTINUE
      X=X+H
      GO TO 1
      CONTINUE
      M=N
      DO 499 II=1,M
      P2(II)=P1(II)
```

```
LL=(X-H)/H
L=FLOAT(I)-LL
CALL VI(Y,C,P1,N)
DO 5001
CONTINUE
X=X+H
GO TO 11
CONTINUE
M=N
DO 4991 II=1,M
P2(II)=P1(II)
CONTINUE
Y=X/H-FLOAT(J)
IF(Y.LT.-1.99) GO TO 8001
IF(Y.GT.2.01) GO TO 8001
LL=(X-H)/H
L=FLOAT(J)-LL
CALL VI(Y,C,P1,N)
CALL POWULT(N,M,P1,P2,MP)
NT=M+N-1
M=NT
DO 5011 IJ=1,M
P2(IJ)=MP(IJ)
CONTINUE
Y=X/H-FLOAT(K)
IF(Y.LT.-1.99) GO TO 8001
IF(Y.GT.2.01) GO TO 8001
LL=(X-H)/H
L=FLOAT(K)-LL
CALL VI(Y,C,P1,N)
CALL POWULT(N,M,P1,P2,MP)
NTO=M+N-1
DO 551 L=1,NTO
CINTEG=CINTEG+MP(L)/L
CONTINUE
B(I,J,K)=CINTEG
CONTINUE
X=X+H
GO TO 11
CONTINUE
CINTEG=0.0;X=0.0
CONTINUE
CINTEG=0.0;X=0.0
CONTINUE
CINTEG=0.0;X=0.0
CONTINUE
RETURN
END
```

THIS PROGRAM COMPUTES THE VALUES OF THE COEFFICIENTS C
IT USES FOUR SUBROUTINES VI , VID , VIDO & POWULT.

```
C IS 39X39 COEFFICIENT MATRIX AND  P1&P2 ARE COEFFICIENTS
OF POLYNOMIALS TO BE MULTIPLIED IN INCREASING POWER OF X.
MP'S ARE COEFFICIENTS OF MULTIPLIED POLYNOMIAL IN INCREASING
POWER OF X.LL STANDS FOR LOWER LIMIT IN EVERY INTEGRATION
INVOLVE).
C                S U B R O U T I N E   C I J N                  C
      SUBROUTINE CIJN(NM,H,CC)
      DIMENSION CC(5,5),P1(10),P2(10)
      REAL MP(20),LL
      CINTEG=0.0;X=0.0
      DO 1002 I=2,NM
      DO 2002 J=2,NM
      IJD=IABS(I-J)
      IF(IJD.GT.3) GO TO 502
      CONTINUE
      Y=X/H-FLOAT(I)
      IF(Y.LT.-1.99) GO TO 302
      IF(Y.GT.2.01) GO TO 502
      LL=(X-H)/H
      N=FLOAT(I)-LL
      CALL VID(Y,C,P1,N)
      GO TO 5002
      CONTINUE
      X=X+H
      GO TO 12
      CONTINUE
      M=N
      DO 4992 II=1,M
      P2(II)=P1(II)
      CONTINUE
      Y=X/H-FLOAT(J)
      IF(Y.LT.-1.99) GO TO 8002
      IF(Y.GT.2.01) GO TO 8002
      LL=(X-H)/H
      N=FLOAT(J)-LL
      CALL VID(Y,C,P1,N)
      CALL POMULT(N,M,P1,P2,MP)
      NT=4*M-1
      DO 552 K=1,NT
      CINTEG=CINTEG+MP(K)/K
      CONTINUE
      CC(I,J)=CINTEG
      CONTINUE
      X=X+H
      GO TO 12
      CONTINUE
      CINTEG=0.0;X=0.0
      CONTINUE
```

```
INTEG=).0;X=0.0
CONTINUE
RETURN
END
------------------------------------------------------------------
FOLLOWING SUBROUTINE VI CALCULATES VALUES OF INTERMEDIATE
FUNCTION VALUES TO BE USED IN CALCULATION OF COEFFICIENTS A,B&C.
------------------------------------------------------------------
!            S U B R O U T I N E   V I                           !
------------------------------------------------------------------
SUBROUTINE VI(Y,C,P1,N)
DIMENSION P1(10)
IF(Y.GT.0.01) GO TO 41
IF(Y.GT.0.01) GO TO 31
IF(Y.GT.-0.99) GO TO 21
N=4
A=2.0-C
P1(1)=(A**3)/6.0
P1(2)=0.6*(A**2)
P1(3)=0.6*A
P1(4)=1.0/6.0
GO TO 101
CONTINUE
N=4
A1=2.0-C
B1=1.0-C
P1(1)=((A1**3)-4.0*(B1**3))/6.0
P1(2)=0.6*(A1**2-4.0*B1**2)
P1(3)=0.6*(A1-4.0*B1)
P1(4)=-0.6
GO TO 101
CONTINUE
N=4
A2=C+2.0
B2=C+1.0
P1(1)=(A2**3-4.0*B2**3)/6.0
P1(2)=0.6*(4.0*B2**2-A2**2)
P1(3)=0.6*(A2-4.0*B2)
P1(4)=0.6
GO TO 101
CONTINUE
N=4
A3=C+2.0
P1(1)=(A3**3)/6.0
P1(2)=-0.5*A3**2
P1(3)=0.6*A3
P1(4)=-1.0/6.0
CONTINUE
RETURN
END
------------------------------------------------------------------
```

```
FOLLOWING SUBROUTINE VID CALCULATES THE COEFFICIENTS
OF DIFFERENTIATED INTERMEDIATE FUNCTIONS
-----------------------------------------------------
!           S U B R O U T I N E   V I D             !
-----------------------------------------------------
SUBROUTINE VID(Y,C,P1,N)
DIMENSION P1(20)
IF(Y.GT.1.01)GO TO 42
IF(Y.GT.0.01) GO TO 32
IF(Y.GT.-0.99) GO TO 22
N=3
A=2.0-C
P1(1)=0.6*(A**2)
P1(2)=A
P1(3)=0.6
GO TO 103
CONTINUE
N=3
A1=2.0-C
B1=1.0-C
P1(1)=0.6*(A1+2.0*B1)*(A1-2.0*B1)
P1(2)=A1-4.0*B1
P1(3)=-1.6
GO TO 103
CONTINUE
N=3
A2=1.0+C
B2=2.0+C
P1(1)=0.6*(2.0*A2+B2)*(2.0*A2-B2)
P1(2)=B2-4.0*A2
P1(3)=1.6
GO TO 103
N=3
A3=2.0+C
P1(1)=-0.5*(A3**2)
P1(2)=A3
P1(3)=-0.5
CONTINUE
RETURN
END
-----------------------------------------------------
FOLLOWING SUBROUTINE VIDD COMPUTES THE COEFFICENTS OF
INTERMEDIATE FUNCTIONS TWICE DIFFERENTIATED
-----------------------------------------------------
!           S U B R O U T I N E   V I D D           !
-----------------------------------------------------
SUBROUTINE VIDD(Y,C,P1,N)
DIMENSION P1(10)
IF(Y.GT.1.01) GO TO 47
IF(Y.GT.0.01) GO TO 37
IF(Y.GT.-0.99) GO TO 27
```

```
N=2
A=2.0-C
P1(1)=A
P1(2)=1.0
GO TO 107
CONTINUE
N=2
A1=2.0-C
B1=1.0-C
P1(1)=A1-4.0*B1
P1(2)=-3.0
GO TO 107
CONTINUE
N=2
A2=2.0+C
B2=1.0+C
P1(1)=A2-4.0*B2
P1(2)=3.0
GO TO 107
CONTINUE
N=2
A3=2.0+C
P1(1)=A3
P1(2)=-1.0
CONTINUE
RETURN
END
C-----------------------------------------------------------------
C NEXT SUBROUTINE FOLLOWS WHICH MULTIPLIES TWO POLYNOMIALS
C P1 & P2 AND GIVES THE COEFFIENTS OF MULTIPLIED POLYNOMIALS
C AS MP'S.
C P1 ARE COEFFICIENTS OF FIRST POLYNOMIAL & P2 ARE COEFFICIENTS
C OF THE SECOND POLYNOMIAL.
C M DENOTES TOTAL NUMBER OF TERMS IN NEW POLYNOMIAL
C-----------------------------------------------------------------
C                S U B R O U T I N E   P O M U L T
C-----------------------------------------------------------------
      SUBROUTINE POMULT(N,M,P1,P2,MP)
      DIMENSION P1(20),P2(20)
      REAL MP(20)
      K=0
      MT=M+N-1
      DO 51 IZ=1,MT
      MP(IZ)=0.0
      CONTINUE
      DO 102 J=1,M
      DO 202 I=1,N
      MP(I+K)=MP(I+K)+P2(J)*P1(I)
      CONTINUE
      K=K+1
      CONTINUE
```

```
RETURN
END
C-------------------------------------------------------------------
C FOLLOWING SUBROUTINE COMPUTES VALUES OF SECH SQR TO BE USED
C FOR COMPUTATION OF INITIAL VALUES OF ALPHA'S.
C-------------------------------------------------------------------
C                 S U B R O U T I N E   S E K                     C
C-------------------------------------------------------------------
      SUBROUTINE SEK(N,H,FO,FOD)
      REAL FO(10),FOD(10)
      C=0.3
      EPS=4.A1E-4
      K=0.5*SQRT(C/EPS)
      D=-K
      X=2.0*H
      DO 111 I=2,N-1
      V=K*X+D
      CV=COSH(V)
      SV=SINH(V)
      FO(I)=3.0*C/(CV*CV)
      FOD(I)=-K6.0*C*K*SV)/(CV*CV*CV)
      X=X+H
      CONTINUE
      RETURN
      END
```

```
@@@
```

```
C@@@
```

```
      Y=(X/H-FLOAT(I))
      IF(Y.GT.0.99) GO TO 31
      IF(Y.GT.-0.01) GO TO 21
      IF(Y.GT.-1.01) GO TO 11
      F1=3.0*C*((X/H-FLOAT(I)+2.0)**3)/(6.0*H*(COSH(RK*X+D))**2)
      GO TO 5)
   11 CONTINUE
      F1=3.0*C*((X/H-FLOAT(I)+2.0)**3-4.0*(X/H-FLOAT(I)+1.0)**3)/
     1  (6.0*H*(COSH(RK*X+D))**2)
      GO TO 5)
   21 CONTINUE
      F1=3.0*C*((-(X/H-FLOAT(I))+2.0)**3-4.0*(-(X/H-FLOAT(I))+1.0)**3)/
     1  (6.0*H*(COSH(RK*X+D))**2)
      GO TO 5)
   31 CONTINUE
      F1=3.0*C*((-(X/H-FLOAT(I))+2.0)**3)/(6.0*H*(COSH(RK*X+D))**2)
    5 RETURN
      END
C-----------------------------------------------------------------------
C FOLLOWING SUBROUTINE ALPHA2 CALCULATES THE INITIAL VALUES OF
C ALPHA FOR TWO SOLITON SOLUTION.
C-----------------------------------------------------------------------
C                  S U B R O U T I N E   A L P H A 2                   I
C-----------------------------------------------------------------------
      SUBROUTINE ALPHA2(N,C1,C2,HC,EPS,AER,RER,BIN)
      REAL DCADRE,F2,A,B,AER,RER,ERR
      REAL BIN(100)
      INTEGER IER
      REAL RK1,RK2,C1,C2
      COMMON I,H,C1,C2,RK1,RK2,D1,D2
      EXTERNAL F2
      H=HC
      C1=EPS
      C2=EPS
      RK1=0.5*SQRT(C1/EPS)
      RK2=0.5*SQRT(C2/EPS)
      D1=-5.0
      D2=-5.0
      DO 101 II=2,N
      I=II
      X=2.0*H
      CONTINUE
      A=X
      B=A+H
      Y=(X/H-FLOAT(I))
      IF(Y.LT.-2.01) GO TO 51
      IF(Y.GT.1.99) GO TO 101
      CC=DCADRE(F2,A,B,AER,RER,ERR,IER)
      BIN(I)=BIN(I)+CC
      CONTINUE
      X=X+H
```

```
GO TO 121
CONTINUE
RETURN
END
C-----------------------------------------------------------------
REAL FUNCTION F2(X)
REAL RK1,RK2
COMMON I,H,C1,C2,RK1,RK2,D1,D2
Y=(X/H-FLOAT(I))
IF(Y.GT.0.99) GO TO 311
IF(X.GT.-0.01) GO TO 211
IF(Y.GT.-1.01) GO TO 111
F2=(3.0*(C1/(COSH(RK1*X+D1))**2+C2/(COSH(RK2*X+D2))**2))*((X/H
1-FLOAT(I)+2.0)**3)/6.0*H
GO TO 501
CONTINUE
F2=(3.0*(C1/(COSH(RK1*X+D1))**2+C2/(COSH(RK2*X+D2))**2))*((X/H
1-FLOAT(I)+2.0)**3-4.0*(X/H-FLOAT(I)+1.0)**3)/6.0*H
GO TO 501
CONTINUE
F2=(3.0*(C1/(COSH(RK1*X+D1))**2+C2/(COSH(RK2*X+D2))**2))*((-(X/H
1-FLOAT(I))+2.0)**3-4.0*(-(X/H-FLOAT(I))+1.0)**3)/6.0*H
GO TO 501
CONTINUE
F2=(3.0*(C1/(COSH(RK1*X+D1))**2+C2/(COSH(RK2*X+D2))**2))*((-(X/H
1-FLOAT(I))+2.0)**3)/6.0*H
RETURN
END
```

```
C
C              S U R F I V E . F O R                              C
C
C
C THIS SUBROUTINE CALCULATES ACTUAL SOLUTION OF THE FUNCTION
C U(x,t), WHERE U(x,t)=SUM ALPHA(J)(t)*V(J)(x) AT DIFFERENT X.
C IT GETS THE VALUES OF ALPHA AT DIFFERENT TIMES FROM MAIN
C PROGRAM A.FOR.
C-------|---------------------------------------------------------
      SUBROUTINE SOLV(N,H,ALPHA,U)
      REAL V(100),ALPHA(100),U(100)
      X=2.0*H
      DO 111 I=2,N
      U(I)=0.0
      DO 110 J=2,N
      V(J)=0.0
      Y=(X/H-FLOAT(J))
      IF(Y.LT.-1.999) GO TO 50
      IF(Y.GT.2.001) GO TO 110
      IF(Y.GT.1.001) GO TO 31
      IF(Y.GT.0.001) GO TO 21
      IF(Y.GT.-0.999) GO TO 11
      V(J)=((Y+2.0)**3)/6.0
      GO TO 49
      V(J)=((Y+2.0)**3-4.0*(Y+1.0)**3)/6.0
      GO TO 49
      V(J)=((-Y+2.0)**3-4.0*((-Y+1.0)**3))/6.0
      GO TO 49
      V(J)=((-Y+2.0)**3)/6.0
      CONTINUE
      U(I)=U(I)+ALPHA(J)*V(J)
      CONTINUE
      CONTINUE
      X=X+H
      CONTINUE
      RETURN
      END
```

SUBSIX.FOR

SUBROUTINE VMULSF (A,N,B,M,IB,C,IC)

VMULSF — SUBSIX — LIBRARY 2

FUNCTION		- MULTIPLICATION OF A MATRIX STORED IN SYMMETRIC STORAGE MODE BY A FULL MATRIX.
USAGE		- CALL VMULSF(A,N,B,M,IB,C,IC)
PARAMETERS	A	- INPUT VECTOR OF LENGTH N(N+1)/2 CONTAINING AN N BY N SYMMETRIC MATRIX STORED IN SYMMETRIC STORAGE MODE.
	N	- ORDER OF A AND NUMBER OF ROWS IN B. (INPUT)
	B	- N BY M MATRIX IN FULL STORAGE MODE. (INPUT)
	M	- NUMBER OF COLUMNS IN B AND NUMBER OF COLUMNS IN OUTPUT MATRIX C. (INPUT)
	IB	- ROW DIMENSION OF B AS SPECIFIED IN THE MAIN PROGRAM. IB MUST BE GREATER THAN OR EQUAL TO N. (INPUT)
	C	- N BY M MATRIX IN FULL STORAGE MODE CONTAINING THE PRODUCT C = AB. (OUTPUT)
	IC	- ROW DIMENSION OF C AS SPECIFIED IN THE MAIN PROGRAM. IC MUST BE GREATER THAN OR EQUAL TO N. (INPUT)
PRECISION		- SINGLE
LANGUAGE		- FORTRAN

SUBROUTINE VCVTFO (A,N,NC,IA,AA,IAA)

VCVTFO — SUBSIX — LIBRARY 2

FUNCTION		- STORAGE MODE CONVERSION - FULL TO BAND SYMMETRIC.
USAGE		- CALL VCVTFO(A,N,NC,IA,AA,IAA)
PARAMETERS	A	- INPUT. N BY N BAND SYMMETRIC MATRIX STORED IN FULL STORAGE MODE.
	N	- INPUT. ORDER OF MATRIX A.
	NC	- INPUT. NUMBER OF UPPER OR LOWER CODIAGONALS IN MATRICES A AND AA.
	IA	- THE ROW DIMENSION OF A AS SPECIFIED IN THE CALLING PROGRAM. IA MUST BE GREATER THAN OR EQUAL TO N. (INPUT)

```
                AA      - OUTPUT. N BY (NC+1) MATRIX CONTAINING A IN
                            BAND SYMMETRIC STORAGE MODE. INPUT MATRIX
                            A MAY BE USED FOR OUTPUT MATRIX AA IF THE
                            USER SO DESIRES. IN THIS CASE IA MUST
                            EQUAL IAA.
                IAA     - THE ROW DIMENSION OF AA AS SPECIFIED IN THE
                            CALLING PROGRAM. IAA MUST BE GREATER THAN
                            OR EQUAL TO N.(INPUT)
  PRECISION             - SINGLE
  LANGUAGE              - FORTRAN
-------------------------------------------------------------------------
    SUBROUTINE LIN2PB (A,N,NC,IA,AINV,IDGT,D1,D2,WK,IER)
-------------------------------------------------------------------------
-LIN2PB-------S-------LIBRARY 2-----------------------------------------
  FUNCTION              - INVERSION OF A MATRIX - SYMMETRIC BAND
                            STORAGE MODE - HIGH ACCURACY SOLUTION
  USAGE                 - CALL LIN2PB(A,N,NC,IA,AINV,IDGT,D1,D2,WK,IER)
  PARAMETERS    A       - N BY N POSITIVE DEFINITE SYMMETRIC BAND MATRIX
                            TO BE INVERTED. A IS STORED IN SYMMETRIC
                            BAND STORAGE MODE AND THEREFORE HAS
                            DIMENSION N BY (NC+1).(INPUT)
                N       - ORDER OF A AND AINV.(INPUT)
                NC      - NUMBER OF UPPER OR LOWER CO-DIAGONALS IN
                            MATRIX A.(INPUT)
                IA      - ROW DIMENSION OF A AS SPECIFIED IN THE MAIN
                            PROGRAM.(INPUT)
                AINV    - OUTPUT VECTOR OF LENGTH N(N+1)/2 CONTAINING
                            THE N BY N INVERSE OF MATRIX A. AINV IS
                            STORED IN SYMMETRIC STORAGE MODE.
                IDGT    - THE APPROXIMATE NUMBER OF DIGITS IN THE
                            ANSWER WHICH WERE UNCHANGED AFTER
                            IMPROVEMENT. (OUTPUT)
                D1,D2   - COMPONENTS OF THE DETERMINANT OF A.
                            DETERMINANT(A) = D1*2.**D2. (OUTPUT)
                WK      - VECTOR OF LENGTH N*(NC+3) USED AS WORK
                            STORAGE. ON RETURN THE FIRST N*(NC+1)
                            LOCATIONS OF WK WILL CONTAIN THE
                            MATRIX L IN BAND STORAGE MODE WHERE A =
                            L*L-TRANSPOSE (THE DIAGONAL ELEMENTS OF L
                            ARE STORED IN RECIPROCAL FORM).
                IER     - ERROR PARAMETER.
                            TERMINAL ERROR = 128+N.
                              N = 1 INDICATES THAT THE MATRIX A IS
                                ALGORITHMICALLY NOT POSITIVE DEFINITE.
                                (SEE THE CHAPTER L PRELUDE).
                              N = 2 INDICATES THAT ITERATIVE IMPROVEMENT
                                FAILED TO CONVERGE. THE MATRIX A IS TOO
                                ILL-CONDITIONED.
```

```
PRECISION                - SINGLE
REQD IMSL ROUTINES       - LUDAPB,LUELPB,LUREPB,UERTST
LANGUAGE                 - FORTRAN
-------------------------------------------------------------------------------

SUBROUTINE DREBS  (DFN,Y,T,N,JM,IND,JSTART,H,HMIN,EPS,R,S,WK,IER)

-DREBS-------S------LIBRARY 2-------------------------------------------------

FUNCTION                 - FIRST ORDER DIFFERENTIAL EQUATION SOLVER -
                           THE METHOD OF BULIRSCH - STOER FOR
                           DY/DT = F(Y,T)
USAGE                    - CALL DREBS(DFN,Y,T,N,JM,IND,JSTART,H,HMIN,
                           EPS,R,S,WK,IER)
PARAMETERS    DFN        - USER SUPPLIED EXTERNAL SUBROUTINE,
                           DFN(Y,T,N,DY). DFN MUST CALCULATE DY(I) =
                           F(Y(1),Y(2),...,Y(N),T) FOR I=1,2,...,N.
              Y          - ON INPUT Y(1),Y(2),...,Y(N) ARE INITIAL VALUES
                           ON OUTPUT Y(1),Y(2),...,Y(N) CONTAIN AN
                           APPROXIMATE SOLUTION TO Y AT T (AS SET ON
                           OUTPUT).
              T          - T IS THE INDEPENDENT VARIABLE. ON INPUT, T
                           SHOULD CONTAIN THE INITIAL VALUE OF THE
                           INDEPENDENT VARIABLE. ON OUTPUT, T
                           CONTAINS THE UPDATED VALUE OF THE INDEPEN-
                           DENT VARIABLE.
              N          - THE NUMBER OF EQUATIONS IN THE SYSTEM
              JM         - THE MAXIMUM ORDER OF THE RATIONAL APPROX-
                           IMATION. JM MUST BE LESS THAN 7.
              IND        - CONVERGENCE TYPE INDICATOR (INPUT)
                           IND = 1 SPECIFIES THE STANDARD ERROR TEST
                           IND = 2 SPECIFIES THE RELATIVE ERROR TEST
                           IND = 3 SPECIFIES THE ABSOLUTE ERROR TEST
              JSTART     - AN INPUT INDICATOR WITH THE FOLLOWING MEANINGS
                           0 - PERFORM A STEP. THE FIRST STEP
                             MUST BE DONE WITH THIS VALUE OF JSTART SO
                             THAT THE SUBROUTINE CAN INITIALIZE ITSELF.
                           -1 - REPEAT THE LAST STEP WITH A NEW VALUE
                             OF H OR JM. THE INITIAL VALUES OF Y, S,
                             AND T ARE SET TO THE INITIAL VALUES OF Y.
                             S, AND T FROM THE MOST RECENT CALL TO
                             DREBS WITH JSTART = 0.
              H          - ON INPUT, H IS AN INITIAL GUESS FOR THE STEP
                           SIZE.
                           ON OUTPUT, H CONTAINS A SUGGESTED STEP SIZE
                           FOR THE NEXT STEP. THE SUGGESTED VALUE MAY
                           BE LARGER OR SMALLER THAN THE ORIGINAL
                           STEP SIZE.
              HMIN       - HMIN IS THE SMALLEST PERMISSIBLE STEP SIZE.
```

```
                  DREBS WILL DECREASE THE STEP SIZE
                  UNTIL CONVERGENCE CAN BE OBTAINED.
EPS       - ERROR TOLERANCE.
R         - ON OUTPUT, THE N-VECTOR R CONTAINS THE
                  ABSOLUTE ERRORS IN EACH COMPONENT FOR
                  THE CURRENT STEP.
S         - AN N-VECTOR CONTAINING EITHER (1) THE LARGEST
                  VALUE OF EACH Y COMPUTED SINCE THE START OF
                  THE INTEGRATION FOR THE STANDARD ERROR TEST,
                  (2) THE LARGEST VALUE OF EACH Y COMPUTED
                  DURING THE CURRENT STEP BEING TAKEN FOR
                  RELATIVE ERROR TEST, OR (3) THE VALUE 1.0
                  FOR THE ABSOLUTE ERROR TEST.
                  S MUST BE INITIALIZED IN ACCORDANCE WITH
                  THE ERROR TEST SELECTED BEFORE THE FIRST
                  CALL TO DREBS. IF IND = 1 OR 2, S(I) SHOULD
                  EQUAL Y(I) ON THE FIRST CALL. IF IND = 3,
                  S(I) SHOULD EQUAL 1.0.
WK        - WORKING STORAGE OF DIMENSION 29*N
IER       - ERROR INDICATOR
                  TERMINAL ERROR = 128 + N
                    N = 1 INDICATES CONVERGENCE COULD NOT BE
                      OBTAINED WITH CURRENT VALUES OF H AND
                      HMIN. Y, T, AND H HAVE BEEN UPDATED.
                  WARNING ERROR (WITH FIX) = 64 + N
                    N = 2 INDICATES JM IS LESS THAN 1 OR GREATER
                      THAN 6. SET JM = 5.

PRECISION         - SINGLE
REQ'D IMSL ROUTINES - UERTST
LANGUAGE          - FORTRAN
```

APPENDIX C

DETAILED TABLES FOR CALCULATED SOLUTIONS

TABLE C1

TIONS AT T= .000

.30)

SIZE IN SPACE HC= 0.05

SIZE IN TIME H= 0.01

)

13837E-07	0.12408E-06	-0.23813E-06	0.49781E-05	-0.78541E-06
19903E-05	-0.19135E-05	0.98480E-05	0.33803E-05	0.69308E-04
13309E-03	0.65294E-03	0.19249E-02	0.73647E-02	0.24024E-01
09208E-01	0.25333E+00	0.62348E+00	0.90392E+00	0.62348E+00
33E+00	0.83021E-01	0.24024E-01	0.73647E-02	0.19249E-02
291E-03	0.13309E-03	0.69309E-04	0.33803E-05	0.98481E-05
9135E-05	0.19903E-05	-0.78540E-06	0.49779E-05	-0.23808E-06
3399E-06	-0.43510E-07			

GURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN

0-2)+1.

TABLE 52

SOLUTIONS AT T= 0.50

= 0.30

STEP SIZE IN SPACE HC= 0.05

STEP SIZE IN TIME H= 0.01

N= 40

-0.24264E-03	-0.19386E-02	0.45161E-02	-0.35361E-02	-0.20965E-02
0.60815E-02	-0.13604E-02	-0.63799E-02	0.38307E-02	0.57889E-02
-0.69788E-02	-0.17882E-02	0.42796E-02	-0.27839E-02	-0.19939E-02
0.40225E-01	0.91765E-02	0.19814E-01	0.86815E-01	0.25615E+00
0.60629E+00	0.91550E+00	0.61773E+00	0.25339E+00	0.87102E-01
0.28858E-01	-0.13730E-01	0.20320E-01	0.67040E-02	-0.65793E-02
-0.31646E-01	0.36995E-01	0.84306E-02	-0.97651E-02	-0.32956E-01
-0.30995E-01	-0.14029E-01			

E FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN
H,(X-Z)H.

TABLE C3

SOLUTIONS AT T= 1.00
C= 0.30
STEP SIZE IN SPACE HC= 0.05
STEP SIZE IN TIME H= 0.02
N= 40

-0.34991E-02	-0.49675E-02	0.11814E-01	-0.10151E-01	-0.37976E-02
0.69885E-02	0.67167E-02	-0.16772E-01	0.94721E-02	0.80898E-02
-0.97351E-02	-0.13139E-01	0.17707E-01	0.43287E-02	-0.11273E-01
-0.62874E-02	0.18896E-01	0.12408E-01	-0.19141E-01	-0.36562E-02
0.46917E-01	0.96505E-01	0.22936E+00	0.62235E+00	0.91056E+00
0.65219E+00	0.28431E+00	0.52194E-01	0.24927E-01	0.16745E-01
0.85570E-01	-0.52157E-01	0.16839E-01	0.17222E-01	0.13432E-01
-0.75297E-01	0.25501E-01			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN
(H,(N-2)H).

TABLE 24

SOLUTIONS AT T= 1.60
C= 0.30
STEP SIZE IN SPACE HC= 0.05
STEP SIZE IN TIME H= 0.01
N= 40

0.20852E-01	-0.16213E-01	-0.15309E-01	0.15583E-01	0.18474E-01
-0.22487E-01	-0.13149E-01	0.26087E-01	0.54721E-02	-0.24905E-01
-0.46874E-02	0.20498E-01	-0.11860E-01	-0.75746E-02	0.22064E-01
0.76617E-02	-0.35642E-01	0.34541E-02	0.34644E-01	0.51407E-02
-0.40590E-01	0.19831E-01	0.35238E-01	0.12150E-01	0.43833E-01
0.28855E+00	0.62676E+00	0.88619E+00	0.61353E+00	0.28521E+00
0.57793E-01	0.21414E-01	-0.26170E-02	0.26950E-01	-0.13889E-01
-0.58702E-02	-0.50693E-01			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN (2H,(1-2)H).

TABLE C5

SOLUTIONS AT T= 2.00
C= 0.30
STEP SIZE IN SPACE HC= 0.05
STEP SIZE IN TIME H= 0.02
N= 40

-0.25335E-01	0.30885E-01	0.35871E-02	-0.31575E-01	0.46303E-03
0.39587E-01	-0.66773E-02	-0.37801E-01	0.79850E-02	0.17884E-01
0.78308E-02	-0.25341E-01	0.62835E-02	0.52434E-02	0.20574E-01
-0.30539E-01	0.42043E-02	-0.26336E-03	0.33803E-01	-0.13245E-01
0.13251E-02	-0.32333E-01	0.21681E-01	0.11123E-01	0.19171E-01
-0.27035E-01	0.32875E-01	0.92853E-01	0.28055E+00	0.58683E+00
0.91593E+00	0.62959E+00	0.22363E+00	0.85275E-01	0.18508E-01
-0.20112E-01	-0.51521E-01			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN
[24,(1-2)H].

TABLE 26

SOLUTIONS AT T= .000

C= 0.30

STEP SIZE IN SPACE HC= 0.04

STEP SIZE IN TIME H= 0.01

N= 50

0.83584E-09	-0.55302E-09	0.63935E-08	0.15252E-08	0.34158E-07
0.86081E-07	0.20317E-06	0.35282E-06	0.13236E-05	0.28945E-05
0.91215E-05	0.22290E-04	0.64826E-04	0.16709E-03	0.46792E-03
0.12369E-02	0.33989E-02	0.90695E-02	0.24529E-01	0.64516E-01
0.15481E+00	0.37950E+00	0.70865E+00	0.90204E+00	0.70865E+00
0.37950E+00	0.15481E+00	0.64516E-01	0.24529E-01	0.90695E-02
0.33989E-02	0.12369E-02	0.46792E-03	0.16709E-03	0.64826E-04
0.22290E-04	0.91215E-05	0.28945E-05	0.13236E-05	0.35281E-06
0.20318E-06	0.85276E-07	0.34167E-07	0.15100E-08	0.64237E-08
-0.61116E-09	0.10532E-08			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN [24,(N-2)4].

TABLE C7

SOLUTIONS AT T= 0.40

C= 0.30

STEP SIZE IN SPACE HC= 0.04

STEP SIZE IN TIME H= 0.02

N= 50

-0.21009E-03	0.46142E-03	-0.20903E-03	0.46326E-03	-0.13634E-03
0.85903E-04	-0.72618E-03	0.64177E-03	0.18384E-03	-0.45400E-03
-0.77938E-03	0.54035E-03	0.96677E-03	-0.60943E-03	-0.28402E-03
-0.81550E-03	0.74520E-03	-0.16957E-03	0.16416E-02	0.38946E-02
0.40565E-01	0.23528E-01	0.64646E-01	0.16509E+00	0.37901E+00
0.70479E+00	0.90582E+00	0.70749E+00	0.37934E+00	0.16478E+00
0.67259E-01	0.22016E-01	0.75456E-02	0.43110E-02	0.66949E-02
-0.38744E-02	-0.85395E-02	-0.84588E-03	0.10518E-01	-0.11826E-02
-0.62583E-02	-0.85055E-02	0.88820E-02	0.36575E-02	0.11537E-02
-0.83821E-02	0.32748E-02			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN [2H,(N-2)H].

TABLE C8

SOLUTIONS AT T= 0.80

C= 0.30

STEP SIZE IN SPACE HC= 0.04

STEP SIZE IN TIME H= 0.00

N= 50

0.11117E-02	-0.78583E-04	-0.16662E-02	0.43071E-02	0.10236E-02
-0.74363E-03	-0.24267E-02	0.16314E-02	0.18342E-02	-0.97438E-03
-0.25952E-02	0.19930E-02	0.21203E-02	-0.12205E-02	-0.42243E-02
0.95098E-03	0.45785E-02	0.10960E-02	-0.59150E-02	-0.94371E-03
0.70942E-02	0.60332E-02	-0.29353E-02	0.34421E-02	0.27716E-01
0.73133E-01	0.15187E+00	0.37079E+00	0.70810E+00	0.91128E+00
0.69860E+00	0.38319E+00	0.16842E+00	0.64947E-01	0.17086E-01
0.44871E-01	0.74164E-02	0.16110E-02	-0.10893E-01	0.20304E-02
0.45385E-02	0.61912E-02	-0.98840E-02	-0.36124E-03	0.41737E-02
0.44575E-01	-0.73809E-02			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN [2H,(1-2H)].

TABLE 39

SOLUTIONS AT T= 1.20
C= 0.30
STEP SIZE IN SPACE HC= 0.04
STEP SIZE IN TIME H= 0.01
N= 50

-0.98263E-03	0.79702E-02	-0.17919E-02	-0.47559E-02	-0.51511E-02
0.97561E-02	0.27160E-02	-0.38644E-02	-0.90315E-02	0.66046E-02
0.60520E-02	0.10158E-03	-0.10978E-01	0.13095E-02	0.56327E-02
0.87390E-02	-0.81609E-02	0.72984E-03	0.49477E-02	0.27770E-02
-0.82831E-02	0.11563E-03	0.65387E-02	0.64202E-02	-0.54447E-02
-0.25525E-02	0.10850E-01	0.32027E-01	0.54241E-01	0.15869E+00
0.87732E+00	0.71053E+00	0.90181E+00	0.70254E+00	0.39094E+00
0.16310E+00	0.61642E-01	0.18694E-01	0.17014E-01	-0.68262E-03
0.58802E-03	-0.80836E-02	0.14149E-01	-0.24655E-02	-0.86116E-02
-0.93553E-02	0.17256E-01			

THE FIGURES IN THE ABOVE TABLE DENOTE THE VALUES OF U(X,T) BETWEEN [2H,(N-2)H].

REFERENCES

[1] LAMB, G.L. : Elements of Soliton Theory
(John Wiley & Sons, 1980).

[2] BHATNAGAR, P.L. : Nonlinear Waves in One-Dimensi
Dispersive Systems
(Oxford University Press, 1979).

[3] WITHAM, G.B. : Linear and Nonlinear Waves
(John Wiley & Sons, 1974).

[4] DODD, R.K., et al : Solitons and Nonlinear Wave E
(Academic Press, 1982).

[5] CHEN, F.F. : Introduction to Plasma Physics and
Controlled Fusion
(Plenum Press, 1984).

[6] DAVIDSON, R.C. : Methods in Nonlinear Plasma Theo
(Academic Press, 1972).

[7] DEBANATH, H.L. : Advances in Nonlinear Waves
(Boston, Pitman, 1985).

[8] ZABUSKY, N.J. and KRUSKAL, M.D. : Interaction of
Solitons in a Collisionless Plasma and the Recurre
of Initial States
(Phys . Rev. Lett. 15, 1965, 240-243).

[9] CALOGERO, F. and DEGASPERIS, A. : Spectral Transf
and Solitons I
(North-Holland, 1982)

[10] LOGREN, K. and SCOTT, A. : Solitons in Action
(Academic Press, 1978).

[11] BULLOUGH, R.K. and CAUDREY, P.J. Solitons
(Springer Verlag, 1980).

[12] GARDNER, G.S. et al.
(Phys, Rev. Lett. 19, 1967).

[13] NOVIKOV, S. et al : Theory of Solitons
(New York Consultant Bureau, 1984).

[14] GRIEG, I.S. and MORRIS, J. L
(J. Comp. Phys. 20, 1976, 60-84).

[15] WAHLBIN, L. : C de Boor, Ed.
(Academic Press, 1974).

[16] ALEXANDER, M.E. and MORRIS, J.K.
(J. Comp. Phys. 30, 1979, 428-451).

[17] MITCHELL, A.R. and SCHOOMBIE, S.W. : Numerical Method
in Coupled Systems
(John Wiley & Sons, 1984).

[18] SANZ-SERNA, J.M. and CHRISTIE, I.
(J. Comp. Phys. 39, 1981, 94-102.

[19] MIURA, R.M. : The KdV Equation: A Survey of Results
(SIAM REV. 18 No.3, 1976).

[20] DRAZIN, P.G. : SOLITONS
(Cambridge University Press, 1983).

[21] ABLOWITZ, M. and SEGUR, H. : Solitons and Inverse
Scattering Transform
(SIAM, Philadelphia, 1981).

[22] ZIENKIWICZ, O.C. and MORGAN, K. : Finite Elements and Approximations
(John Wiley & Sons, 1983).

[23] REDDI, J.N.: An Introduction to the Finite Element Method
(McGraw-Hill, 1984).

[24] MITCHELL, A.R. and WAIT, R. : Finite Element Method i PDE
(John Wiley & Sons, 1977).

[25] FORNBERG, B. and WITHAM, G.B.
(Phil. Trans. Roy. Soc., 289, 1978, 373-404).

[26] ANDERSSEN, R.S., and MITCHELL, A.R.
(Math. Meth. Appl. Sci. 1, 1979, 3-15).

[27] A Practical Guide to Splines:
(Carl de Boor Eds., 1978).

[28] KING, J.T. : Introduction to Numerical Computations
(McGraw Hill, 1984).

NET- 1987 - M - HAS - NUM